Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 24 October 2026

A. Kario
Red Hat, Inc.

22 April 2026

GSS-API Key Exchange with hybrid ML-KEM
draft-kario-gss-keyex-pqc-00

Abstract

This document specifies additions to RFC4462. It defines a new key exchange methods that use hybrid Post-Quantum Traditional (PQ/T) key exchange. The purpose of this specification is to modernize the cryptographic primitives used by Generic Security Service (GSS) key exchanges.

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 24 October 2026.

Table of Contents

1. Introduction

Secure Shell (SSH) Generic Security Service Application Program Interface (GSS-API) methods [RFC4462] allow the use of GSS-API [RFC2743] for authentication and key exchange in SSH. This document updates [RFC4462] with new methods based on [I-D.ietf-sshm-mlkem-hybrid-kex] intended to support environments that desire to use key exchanges resistant to attacks by CRQC (Cryptographically Relevant Quantum Computers).

2. Rationale

As documented in [I-D.ietf-sshm-mlkem-hybrid-kex] traditional cryptography (Finite Field Diffie-Hellman and Elliptic Curve Diffie-Hellman) will not be secure agains CRQCs, to address that we propose use of hybrid Post-Quantum Traditional (PQ/T) cryptography together with GSS-API methods.

3. Document Conventions

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.

4. New PQ/T Hybrid Key Exchange methods

In [I-D.ietf-sshm-mlkem-hybrid-kex] new SSH key exchange algorithms based on PQ/T Hybrid Key Exchange Methods are introduced. We reuse much of section 2 of [I-D.ietf-sshm-mlkem-hybrid-kex] to define GSS-API-authenticated PQ/T Hybrid Key Exchanges.

4.1. Generic GSS-API Key Exchange with PQ/T Hybrid

This section reuses much of the scheme defined in Section 2.1 of [RFC4462] and combines it with the scheme defined in Section 2 of [I-D.ietf-sshm-mlkem-hybrid-kex]; in particular, all checks and verification steps prescribed in Section 2.1 of [I-D.ietf-sshm-mlkem-hybrid-kex] apply here as well.

This section defers to [RFC7546] as the source of information on GSS-API context establishment operations, Section 3 being the most relevant. All Security Considerations described in [RFC7546] apply here too.

A GSS Context is established according to Section 4 of [RFC5656]; The client initiates the establishment using GSS_Init_sec_context() and the server responds to it using GSS_Accept_sec_context(). For the negotiation, the client MUST set mutual_req_flag and integ_req_flag to "true". In addition, deleg_req_flag MAY be set to "true" to request access delegation, if requested by the user. Since the key exchange process authenticates only the host, the setting of anon_req_flag is immaterial to this process. If the client does not support the "gssapi-keyex" user authentication method described in Section 4 of [RFC4462], or does not intend to use that method in conjunction with the GSS-API context established during key exchange, then anon_req_flag SHOULD be set to "true". Otherwise, this flag MAY be set to true if the client wishes to hide its identity. This key exchange process will exchange only a single message token once the context has been established, therefore the replay_det_req_flag and sequence_req_flag SHOULD be set to "false".

The client MUST include its Traditional public key and Post-Quantum encapsulation key with the first message it sends to the server during this process; if the server receives more than one key or none at all, the key exchange MUST fail. That is, the Q_C field of SSH_MSG_KEXGSS_INIT must contain the concatenation of C_PK2 and C_PK1 from section 2.1 of [I-D.ietf-sshm-mlkem-hybrid-kex].

During GSS Context establishment multiple tokens may be exchanged by the client and the server. When the GSS Context is established (major_status is GSS_S_COMPLETE) the parties check that mutual_state and integ_avail are both "true". If not the key exchange MUST fail.

When the GSS Context is established, the Q_S field in server's SSH_MSG_KEXGSS_COMPLETE message needs to contain the concatenation of of S_CT2 and S_PK1 from section 2.1 of [I-D.ietf-sshm-mlkem-hybrid-kex].

Once a party receives the peer's public key it proceeds to compute a shared secret K. This is done as specified in section 2.4 of [I-D.ietf-sshm-mlkem-hybrid-kex].

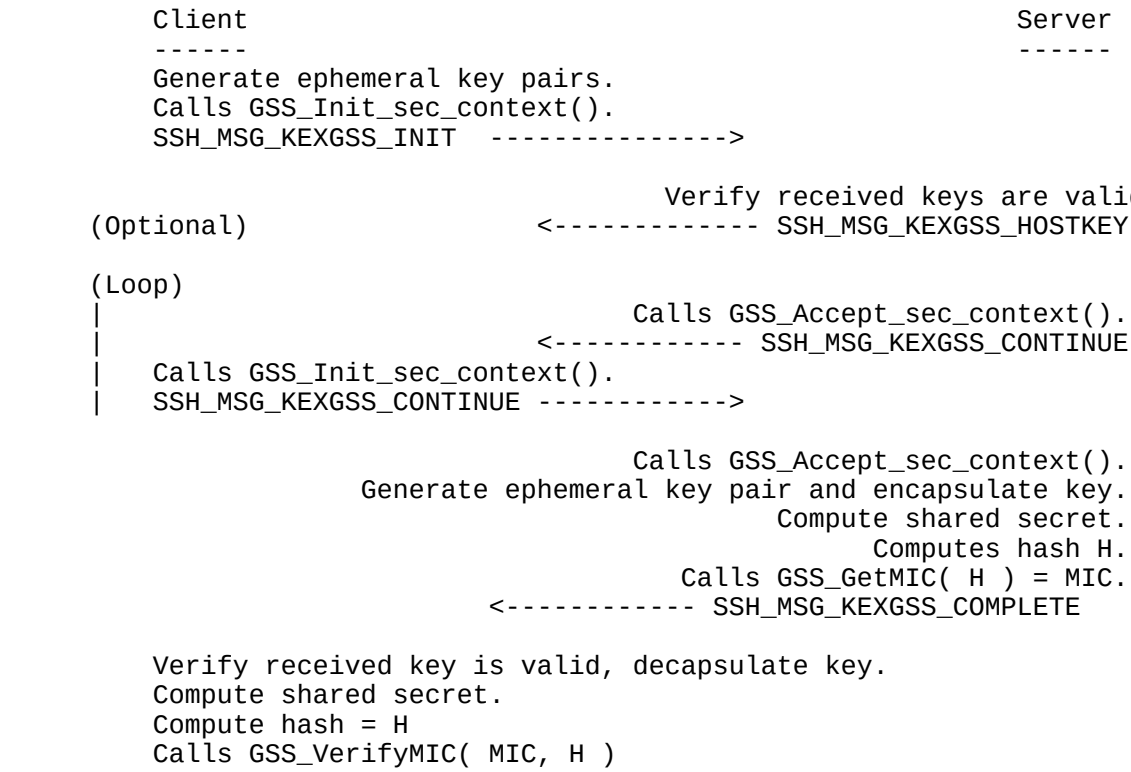
To verify the integrity of the handshake, peers use the Hash Function defined by the selected Key Exchange method to calculate H:

H = hash(V_C || V_S || I_C || I_S || K_S || Q_C || Q_S || K).

The GSS_GetMIC() call is used by the server with H as the payload and generates a MIC. The GSS_VerifyMIC() call is used by the client to verify the MIC.

If any GSS_Init_sec_context() or GSS_Accept_sec_context() returns a major_status other than GSS_S_COMPLETE or GSS_S_CONTINUE_NEEDED, or any other GSS-API call returns a major_status other than GSS_S_COMPLETE, the key exchange MUST fail. The same recommendations expressed in Section 2.1 of [RFC4462] are followed with regards to error reporting.

The following is an overview of the key exchange process:



This is implemented with the following messages:

The client sends:

```
byte      SSH_MSG_KEXGSS_INIT
string    output_token (from GSS_Init_sec_context())
string    Q_C, client's ephemeral public keys octet string
```

The server may responds with:

```
byte      SSH_MSG_KEXGSS_HOSTKEY
string    server public host key and certificates (K_S)
```

The server sends:

```
byte      SSH_MSG_KEXGSS_CONTINUE
string    output_token (from GSS_Accept_sec_context())
```

Each time the client receives the message described above, it makes another call to GSS_Init_sec_context().

The client sends:

```
byte      SSH_MSG_KEXGSS_CONTINUE
string    output_token (from GSS_Init_sec_context())
```

As the final message the server sends either:

```
byte      SSH_MSG_KEXGSS_COMPLETE
string    Q_S, server's ephemeral public key and encapsulated key
          octet string
string    mic_token (MIC of H)
boolean   TRUE
string    output_token (from GSS_Accept_sec_context())
```

Or the following if no output_token is available:

```
byte      SSH_MSG_KEXGSS_COMPLETE
string    Q_S, server's ephemeral public key and encapsulated key
          octet string
string    mic_token (MIC of H)
boolean   FALSE
```

The hash H is computed as the HASH hash of the concatenation of the following:

```
string    V_C, the client's version string (CR, NL excluded)
string    V_S, server's version string (CR, NL excluded)
string    I_C, payload of the client's SSH_MSG_KEXINIT
string    I_S, payload of the server's SSH_MSG_KEXINIT
string    K_S, server's public host key
string    Q_C, client's ephemeral public keys octet string
string    Q_S, server's ephemeral public key and encapsulated key
          octet string
mpint     K,   shared secret
```

This value is called the exchange hash, and it is used to authenticate the key exchange. The exchange hash SHOULD be kept secret. If no SSH_MSG_KEXGSS_HOSTKEY message has been sent by the server or received by the client, then the empty string is used in place of K_S when computing the exchange hash.

Since this key exchange method does not require the host key to be used for any encryption operations, the SSH_MSG_KEXGSS_HOSTKEY message is OPTIONAL. If the "null" host key algorithm described in Section 5 of [RFC4462] is used, this message MUST NOT be sent.

If the client receives a SSH_MSG_KEXGSS_CONTINUE message after a call to GSS_Init_sec_context() has returned a major_status code of GSS_S_COMPLETE, a protocol error has occurred and the key exchange MUST fail.

If the client receives a SSH_MSG_KEXGSS_COMPLETE message and a call to GSS_Init_sec_context() does not result in a major_status code of GSS_S_COMPLETE, a protocol error has occurred and the key exchange MUST fail.

4.2. PQ/T Key Exchange Methods

The following new key exchange methods are defined:

Key Exchange Method Name	Implementation Recommendations
gss-mlkem768nistp256-sha256-*	SHOULD/RECOMMENDED
gss-mlkem1024nistp384-sha384-*	MAY/OPTIONAL
gss-mlkem768x25519-sha256-*	SHOULD/RECOMMENDED

Table 1

Each key exchange method is implicitly registered by this document. The IESG is considered to be the owner of all these key exchange methods; this does NOT imply that the IESG is considered to be the owner of the underlying GSS-API mechanism.

Each method in any family of methods specifies GSS-API-authenticated Post-Quantum Traditional Hybrid key exchanges as described in Section 4.1. The method name for each method is the concatenation of the family method name with the Base64 encoding of the MD5 hash [RFC1321] of the ASN.1 DER encoding [ISO-IEC-8825-1] of the underlying GSS-API mechanism's OID. Base64 encoding is described in Section 6.8 of [RFC2045].

Family method refences

Family Name prefix	Hash Function	Parameters / Function Name	Definition
gss-mlkem768nistp256-sha256-	SHA-256	mlkem768nistp256	Section 2.3.1 of [I-D.ietf-sshm-mlkem-hybrid-kex]

gss-mlkem1024nistp384-sha384-	SHA-384	mlkem1024nistp384	Section 2.3.2 of [I-D.ietf-sshm-mlkem-hybrid-kex]
gss-mlkem768x25519-sha256-	SHA-256	mlkem768x25519	Section 2.3.3 of [I-D.ietf-sshm-mlkem-hybrid-kex]

Table 2

5. IANA Considerations

This document augments the SSH Key Exchange Method Names in [RFC4462].

IANA is requested to update the SSH Protocol Parameters [IANA-KEX-NAMES] registry with the following entries:

Key Exchange Method Name	Reference	OK to Implement
gss-mlkem768nistp256-sha256-*	This draft	SHOULD
gss-mlkem1024nistp384-sha384-*	This draft	SHOULD
gss-mlkem768x25519-sha256-*	This draft	SHOULD

Table 3

6. Security Considerations

6.1. New PQ/T key exchange mechanisms

Although a new cryptographic primitive is used with these methods the actual key exchange closely follows the key exchange defined in [I-D.ietf-sshm-mlkem-hybrid-kex]; therefore all the original Security Considerations as well as those expressed in [I-D.ietf-sshm-mlkem-hybrid-kex] apply.

6.2. GSSAPI Delegation

Some GSSAPI mechanisms can act on a request to delegate credentials to the target host when the deleg_req_flag is set. In this case, extra care must be taken to ensure that the acceptor being authenticated matches the target the user intended. Some mechanisms implementations (like commonly used krb5 libraries) may use insecure DNS resolution to canonicalize the target name; in these cases spoofing a DNS response that points to an attacker-controlled machine

may results in the user silently delegating credentials to the attacker, who can then impersonate the user at will.

7. References

7.1. Normative References

[I-D.ietf-sshm-mlkem-hybrid-kex]
Kampanakis, P., Stebila, D., and T. Hansen, "PQ/T Hybrid Key Exchange with ML-KEM in SSH", Work in Progress, Internet-Draft, draft-ietf-sshm-mlkem-hybrid-kex-10, 26 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-sshm-mlkem-hybrid-kex-10>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[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>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <https://www.rfc-editor.org/info/rfc2743>.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, DOI 10.17487/RFC4462, May 2006, <https://www.rfc-editor.org/info/rfc4462>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[RFC7546] Kaduk, B., "Structure of the Generic Security Service (GSS) Negotiation Loop", RFC 7546, DOI 10.17487/RFC7546, May 2015, <https://www.rfc-editor.org/info/rfc7546>.

[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>.

7.2. Informative References

[IANA-KEX-NAMES]
IANA, "Secure Shell (SSH) Protocol Parameters: Key Exchange Method Names", 2 June 2005, <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16>.

[ISO-IEC-8825-1]
International Organization for Standardization / International Electrotechnical Commission, "ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1, 15 November 2015, <http://standards.iso.org/ittf/PubliclyAvailableStandards/c068345_ISO_IEC_8825-1_2015.zip>.

Author’s Address

Alicja Kario
Red Hat, Inc.
Purkynova 115
612 00 Brno
Czech Republic
Email: hkario@redhat.com