EMU
Internet-Draft
Intended status: Standards Track
Expires: 18 September 2026

T. Reddy
A. Banerjee
Nokia

17 March 2026

Post-Quantum Key Encapsulation Mechanisms (PQ KEMs) in EAP-AKA prime
draft-ietf-emu-pqc-eapaka-02

Abstract

Forward Secrecy for the Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' FS) is specified in [RFC9678], providing updates to [RFC9048] with an optional extension that offers ephemeral key exchange using the traditional Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key agreement algorithm for achieving perfect forward secrecy (PFS). However, it is susceptible to future threats from Cryptographically Relevant Quantum Computers, which could potentially compromise a traditional ephemeral public key. If the adversary has also obtained knowledge of the long-term key and ephemeral public key, it could compromise session keys generated as part of the authentication run in EAP-AKA'.

This draft aims to enhance the security of EAP-AKA' FS protocol by making it quantum-safe using Post-Quantum Key Encapsulation Mechanisms (PQ-KEMs).

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-emu-pqc-eapaka/.

Discussion of this document takes place on the emu Working Group mailing list (mailto:emu@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/emu/. Subscribe at https://www.ietf.org/mailman/listinfo/emu/.

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/.

Copyright Notice

Table of Contents

1. Introduction

Forward Secrecy for the Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' FS) defined in [RFC9678] updates the improved Extensible Authentication Protocol Method for 3GPP Mobile Network Authentication and Key Agreement (EAP-AKA') specified in [RFC9048], with an optional extension providing ephemeral key exchange. This prevents an attacker who has gained access to the long term key from obtaining session keys established in the past, assuming these have been properly deleted. EAP-AKA' FS mitigates passive attacks (e.g., large scale pervasive monitoring) against future sessions.

Nevertheless, EAP-AKA' FS uses traditional algorithms public-key algorithms (e.g., ECDH) which will be broken by a Cryptographically Relevant Quantum Computer (CRQC) using Shor's algorithm. The presence of a CRQC would render state-of-the-art, traditional public-key algorithms deployed today obsolete and insecure, since the assumptions about the intractability of the mathematical problems for these algorithms that offer confident levels of security today no longer apply in the presence of a CRQC. A CRQC could recover the SHARED_SECRET from the ECDHE public keys (Section 6.3 of [RFC9678]). If the adversary has also obtained knowledge of the long-term key, it could then compute CK', IK', and the SHARED_SECRET, and any derived output keys. This means that the CRQC would disable the forward security capability provided by [RFC9678].

Researchers have developed Post-Quantum Key Encapsulation Mechanisms (PQ-KEMs) to provide secure key establishment resistant against an adversary with access to a quantum computer.

At the time of writing, NIST has standardized three PQC algorithms, with more expected to be standardized in the future ([NISTFINAL]). As these algorithms are secure against both quantum and classical computers, this document proposes a PQ-KEM for achieving perfect forward secrecy in EAP-AKA'.

Although the proposed mechanism is applicable to any post-quantum Key Encapsulation Mechanism (PQ-KEM), this document specifies its use with Module-Lattice-based Key Encapsulation Mechanisms (ML-KEMs). ML-KEM provides a one-pass (store-and-forward) cryptographic method for an originator to securely transmit keying material to a recipient using the recipient' s ML-KEM public key. Three parameter sets for

ML-KEM are defined in [FIPS203], namely ML-KEM-512, ML-KEM-768, and ML-KEM-1024, listed in order of increasing security strength and decreasing performance.

2. Conventions and Definitions

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.

3. Terminology

This document makes use of the terms defined in [I-D.ietf-pquip-pqt-hybrid-terminology]. The following terms are repeately used in this specification:

* KEM: Key Encapsulation Mechanism

* PQ-KEM: Post-Quantum Key Encapsulation Mechanism

* CEK: Content Encryption Key

* ML-KEM: Module-Lattice-based Key Encapsulation Mechanism

For the purposes of this document, it is helpful to be able to divide cryptographic algorithms into two classes:

"Asymmetric Traditional Algorithm": An asymmetric cryptographic algorithm based on integer factorisation, finite field discrete logarithms or elliptic curve discrete logarithms, or related mathematical problems.

"Post-Quantum Algorithm": An asymmetric cryptographic algorithm that is believed to be secure against attacks using quantum computers as well as classical computers. Post-quantum algorithms can also be called quantum-resistant or quantum-safe algorithms. Examples of Post-Quantum Algorithm include ML-KEM.

4. Background on EAP-AKA’ with perfect forward secrecy

In EAP-AKA’, The authentication vector (AV) contains a random part RAND, an authenticator part AUTN used for authenticating the network to the USIM, an expected result part XRES, a 128-bit session key for integrity check IK, and a 128-bit session key for encryption CK.

As described in the draft [RFC9678], the server has the EAP identity of the peer. The server asks the AD to run AKA algorithm to generate RAND, AUTN, XRES, CK and IK. Further it also derives CK' and IK' keys which are tied to a particular network name. The server now generates the ephemeral key pair and sends the public key of that key pair and the first EAP method message to the peer. In this EAP message, AT_PUB_ECDHE (carries public key) and the AT_KDF_FS(carries other FS related parameters). Both of these might be ignored if USIM doesn' t support the Forward Secrecy extension. The peer checks if it wants to have a Forward extension in EAP AKA'. If yes, then it will eventually respond with AT_PUB_ECDHE and MAC. If not, it will ignore AT_PUB_ECDHE. If the peer wants to participate in FS extension, it will then generate its ECDH key pair, calculate a shared key based on its private key and server public key. The server will receive the RES from peer and AT_PUB_ECDHE. The shared key will be generated both in the peer and the server with key pairs exchanged, and later master key is also generated.

MK_ECDHE = PRF'(IK'| CK'|SHARED_SECRET,"EAP-AKA' FS"|Identity)

4.1. Key Encapsulation Mechanisms

For the purposes of this document, we consider a Key Encapsulation Mechanism (KEM) to be any asymmetric cryptographic scheme comprised of algorithms satisfying the following interfaces [PQCAPI].

* def kemKeyGen() -> (pk, sk)

* def kemEncaps(pk) -> (ct, ss)

* def kemDecaps(ct, sk) -> ss

where pk is public key, sk is secret key, ct is the ciphertext representing an encapsulated key, and ss is shared secret.

KEMs are typically used in cases where two parties, hereby refereed to as the "encapsulater" and the "decapsulater", wish to establish a shared secret via public key cryptography, where the decapsulater has an asymmetric key pair and has previously shared the public key with the encapsulater.

5. Design Rationales

It is essential to note that in the PQ-KEM, one needs to apply Fujisaki-Okamoto [FO] transform or its variant [HHK] on the PQC KEM part to ensure that the overall scheme is IND-CCA2 secure, as mentioned in [I-D.ietf-tls-hybrid-design]. The FO transform is performed using the KDF such that the PQC KEM shared secret achieved is IND-CCA2 secure.

Note that during the transition from traditional to post-quantum algorithms, there may be a desire or a requirement for protocols that incorporate both types of algorithms until the post-quantum algorithms are fully trusted. HPKE is an KEM that can be extended to support hybrid post-quantum KEMs and the specifications for the use of HPKE with EAP-AKA prime is described in [I-D.draft-ar-emu-pqc-eapaka].

6. PQC KEM Enhancements by Design

We suggest the following changes and enhancements:

* A new attribute, AT_PUB_KEM, is defined to carry the PQC KEM public key from the EAP server.

* A new attribute, AT_KEM_CT, is defined to carry the ciphertext (ct) generated by the PQC KEM Encapsulation function from the EAP peer.

* The AT_KDF_FS attribute is updated to indicate the PQC KEM for generating the Master Key MK_PQ_SHARED_SECRET.

* Multiple AT_KDF_FS attributes are included in the EAP-Request to handle the EAP peer not supporting PQC KEM.

* The PQC KEM can be included first in the AT_KDF_FS attribute in the EAP-Request to indicate a higher priority for its use compared to the traditional key derivation functions.

* EAP-AKA and EAP-AKA′ FS do not define a native attribute-level fragmentation mechanism. As PQC public keys and ciphertexts may exceed the EAP MTU, this specification defines an explicit attribute-level fragmentation mechanism (AT_FRAGMENT) to support transport of large attribute values.

7. Message Fragmentation and Reassembly

EAP-AKA and EAP-AKA' FS do not natively define fragmentation. This specification therefore defines attribute-level fragmentation for attributes whose total length may exceed the EAP MTU.

This specification defines an attribute-level fragmentation mechanism similar to the lock-step acknowledgment model used by EAP-TLS [RFC2716]. Fragmentation applies to the entire attribute, including the attribute header and value.

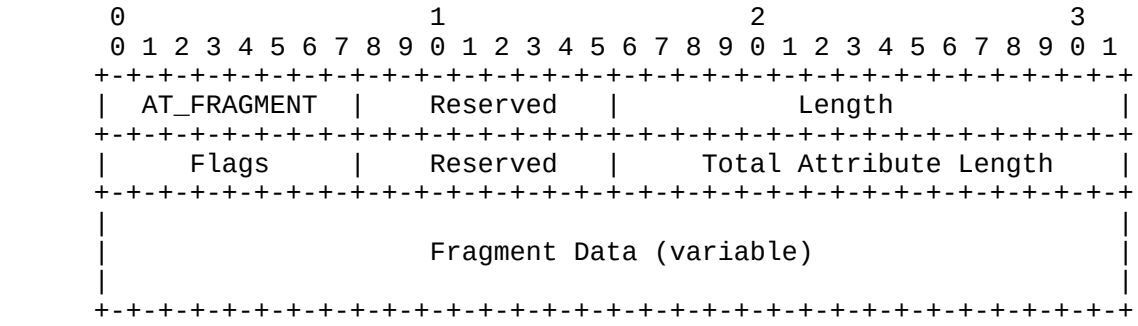
Only one fragmented attribute exchange (i.e., one fragmented attribute transmission in progress in either direction) MUST be active at any time.

7.1. Fragmentation Attribute

When an attribute is fragmented, each fragment carries a Fragmentation attribute. The Fragment Data field carries a contiguous portion of the original attribute, treated as an opaque sequence of octets. The first fragment MUST begin with the attribute Type and Length fields. Subsequent fragments carry the next contiguous octets of the attribute.

The receiver MUST reconstruct the original attribute by concatenating the Fragment Data fields, in the order received, excluding any per-fragment alignment padding. The reassembled attribute MUST be bitwise identical to the original, unfragmented attribute and MUST NOT be processed until reassembly has completed.

The Fragmentation attribute has the following format:



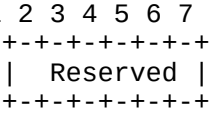
Length:

A 2-octet unsigned integer indicating the length of this AT_FRAGMENT attribute in multiples of 4 octets, including the Type, Reserved, Length, Flags, Total Attribute Length, Fragment Data, and any alignment padding. The total length of the attribute in octets is obtained by multiplying this field by 4.

Each AT_FRAGMENT attribute MUST have a Length that is a multiple of 4 octets.

Flags (1 octet):

The Flags field contains the following bits:



* S (First Fragment)

  Indicates that this is the first fragment of a fragmented attribute. This bit MUST be set on the first fragment and MUST NOT be set on subsequent fragments.

* M (More Fragments) Indicates that additional fragments follow. This bit MUST be set on all fragments except the last.

* Reserved
  MUST be set to zero on transmission and ignored on receipt.

Total Attribute Length (2 octets)

The Total Attribute Length field specifies the total length, in octets, of the unfragmented attribute, including its Type, Length, and Value fields. It is encoded as an unsigned 16-bit integer in network byte order.

This field MUST be present in all fragments belonging to the same fragmented attribute. In fragments other than the first, the value of Total Attribute Length MUST be identical to that of the first fragment. If a mismatch is detected, the receiver MUST treat this as a protocol error and abort the authentication exchange.

7.2. Fragmentation Procedure

* When an EAP peer receives an EAP-Request containing an attribute fragment with the M bit set, it MUST respond with an EAP-Response of the same EAP type containing no attributes. This response serves as a fragment acknowledgment.

* When an EAP server receives an EAP-Response containing an attribute fragment with the M bit set, it MUST respond with an EAP-Request of the same EAP type containing no attributes. This request serves as a fragment acknowledgment.

* The sender MUST NOT transmit the next fragment until the corresponding acknowledgment has been received.

7.3. Use of the EAP Identifier

The EAP Identifier field is used to correlate fragments and acknowledgments:

* The Identifier in an EAP-Response MUST match the Identifier of the immediately preceding EAP-Request.

* Fragment acknowledgments MUST echo the Identifier of the fragment being acknowledged.

* Retransmitted fragments MUST reuse the same Identifier value as the original transmission.

* For fragmented exchanges initiated by the EAP server, the Identifier in each EAP-Request carrying a fragment MUST be incremented relative to the previous EAP-Request.

7.4. Reassembly

The receiver MUST reassemble attribute fragments strictly in the order received and MUST NOT process the fragmented attribute until all fragments have been successfully received and validated.

The final fragment is identified by the M bit being cleared (M = 0). The receiver MUST acknowledge each fragment, including the final fragment, using the lock-step procedure defined for fragmentation. The sender MUST wait for acknowledgment of the final fragment before considering the fragmented attribute exchange complete.

During the fragmentation phase (i.e., while the M bit is set in AT_FRAGMENT), the EAP peer MUST respond to each EAP-Request/AKA'-Challenge fragment with an EAP-Response/AKA'-Challenge message

containing a zero-length attribute payload. These responses serve solely as transport-level acknowledgments and MUST NOT trigger any AKA' cryptographic processing.

The EAP peer MUST NOT initiate USIM processing (e.g., passing RAND and AUTN to the USIM) while attribute fragmentation is in progress. USIM processing has to occur only after the final fragment (M = 0) has been received and the complete attribute set has been successfully reassembled and validated. At that point, the peer will invoke the AKA' algorithm using the RAND and AUTN values contained in the reassembled message.

If a fragment is lost or corrupted, normal EAP retransmission procedures apply. Retransmitted fragments MUST use the same EAP Identifier value as the original transmission.

The receiver MUST verify that:

* The first fragment has the S bit set and subsequent fragments do not; and

* The cumulative length of all received Fragment Data fields equals the Total Attribute Length.

Any inconsistency in fragmentation state (including unexpected S bit usage, receipt of a new initial fragment while reassembly is in progress, length mismatch, or malformed sequencing) MUST be treated as a protocol error, and the authentication exchange MUST be aborted. If reassembly cannot be successfully completed after a bounded number of retransmissions, the authentication exchange MUST be aborted.

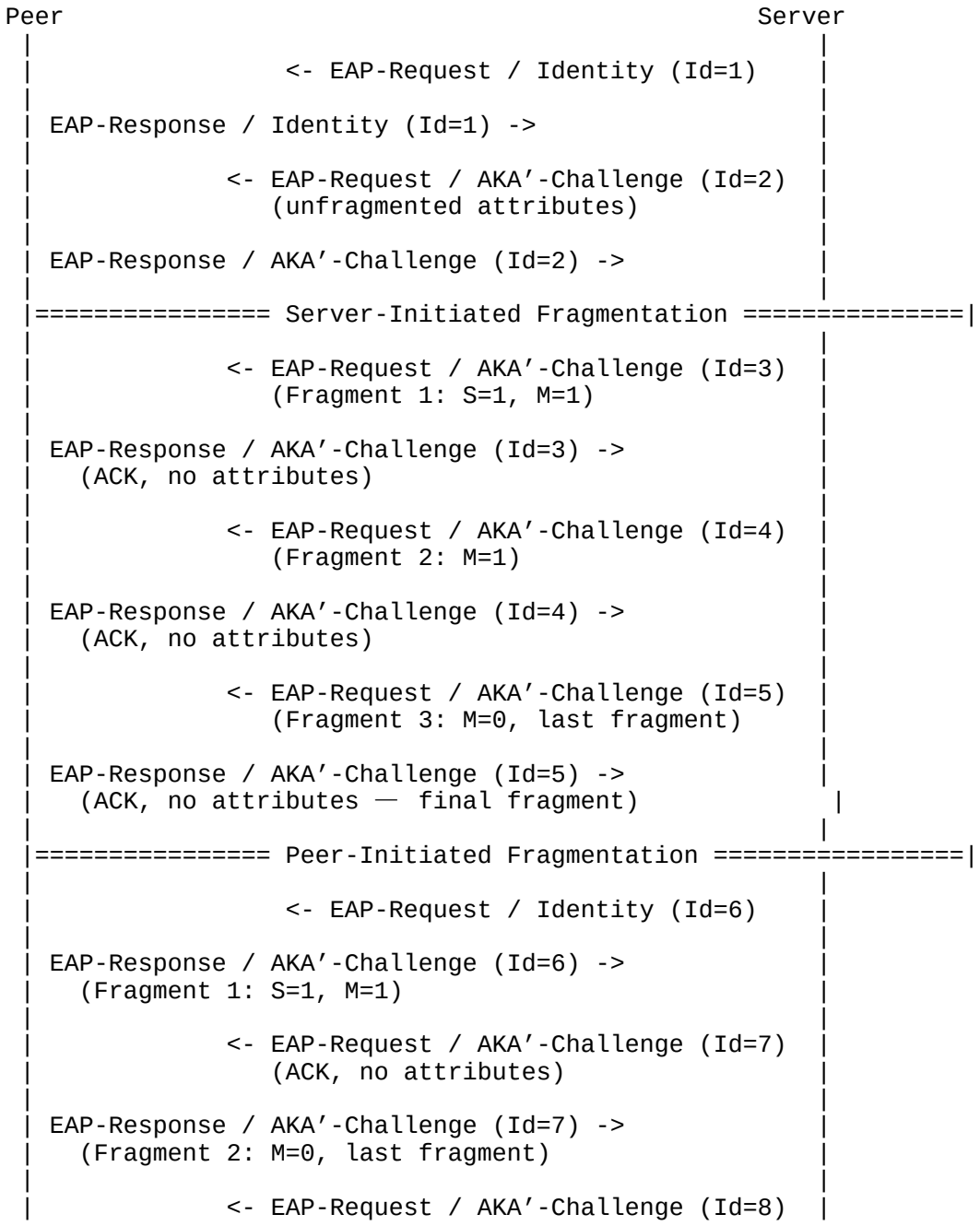
Fragmentation does not modify the AT_MAC calculation rules defined in [RFC9048]. AT_MAC is calculated over the EAP packet exactly as transmitted on the wire, including any AT_FRAGMENT attributes.

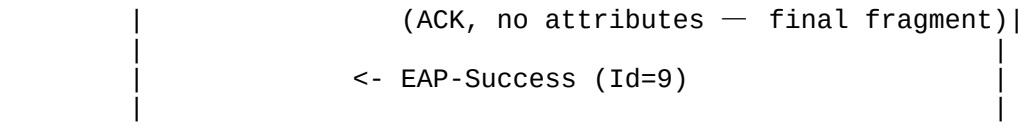
Processing of data contained in reassembled fragmented attributes MUST occur only after successful AT_MAC verification. Fragmentation therefore does not alter the integrity protection scope of the EAP packet.

7.5. Applicability

This fragmentation mechanism applies to any attribute within this EAP method whose encoded length exceeds the EAP MTU. (e.g., AT_PUB_KEM, AT_KEM_CT, or AT_IDENTITY carrying a large SUCI). Attributes that fit within a single EAP packet MUST be sent unfragmented.

The following diagram details how the fragmentation works for both request and response:



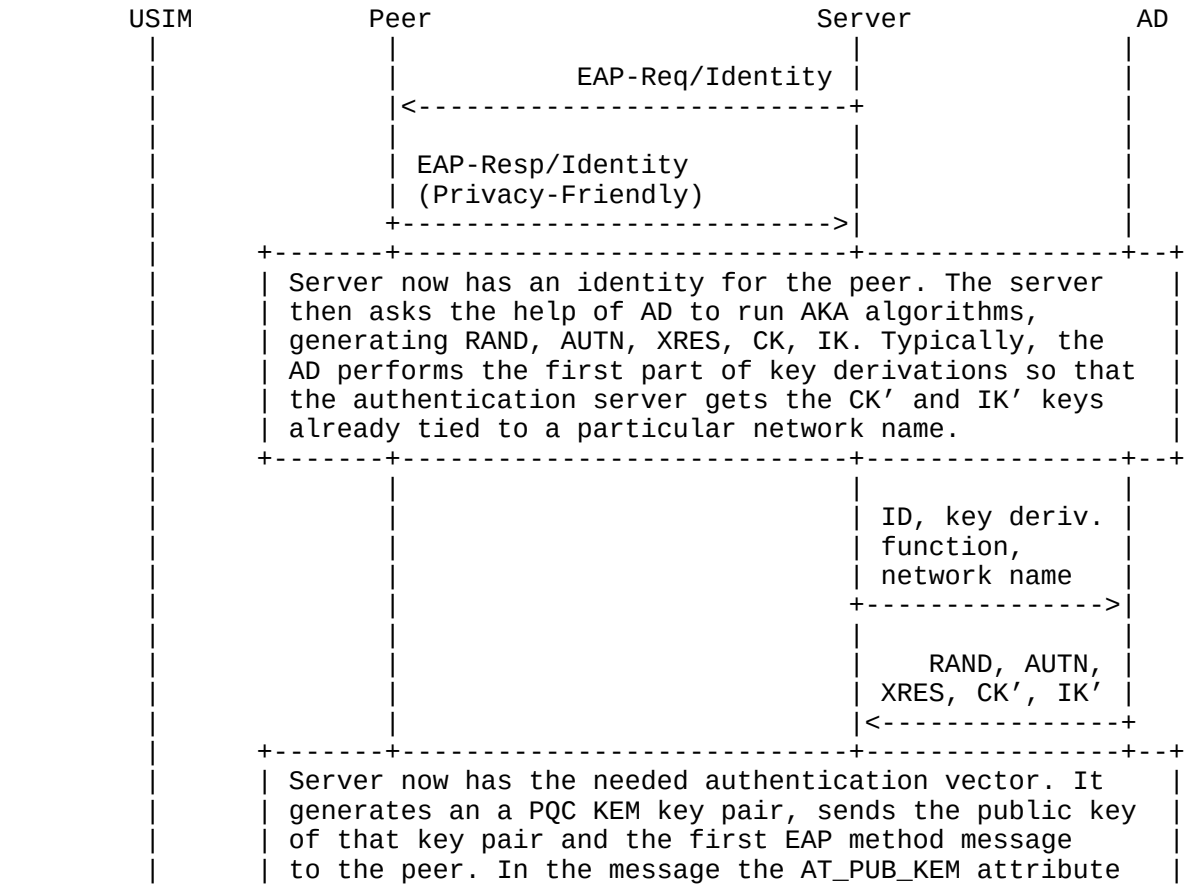


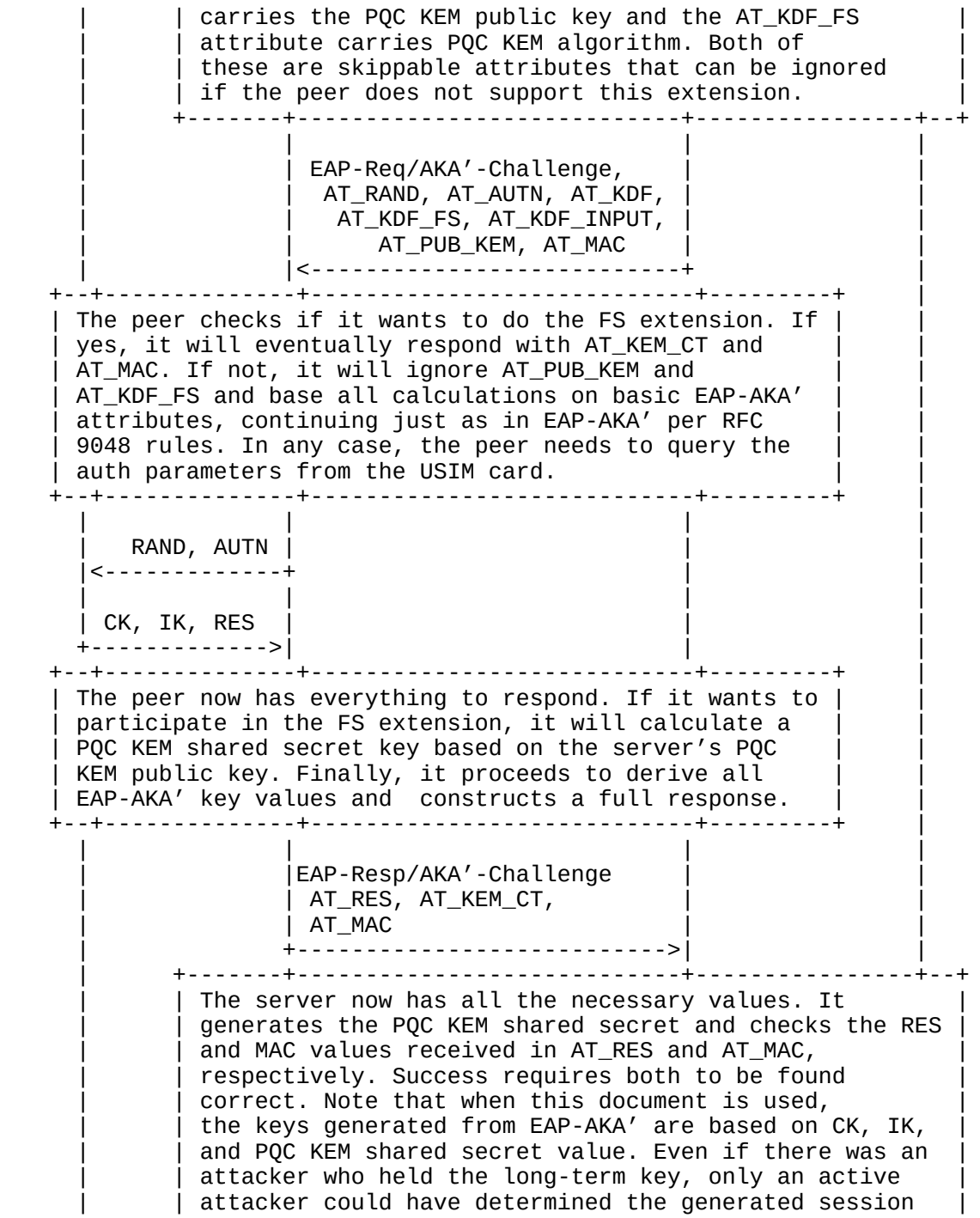
The term “ACK” in the above figure is used for illustrative purpose to describe an EAP-Request or EAP-Response of the same EAP method type that contains no attributes and is sent solely to acknowledge receipt of a fragment.

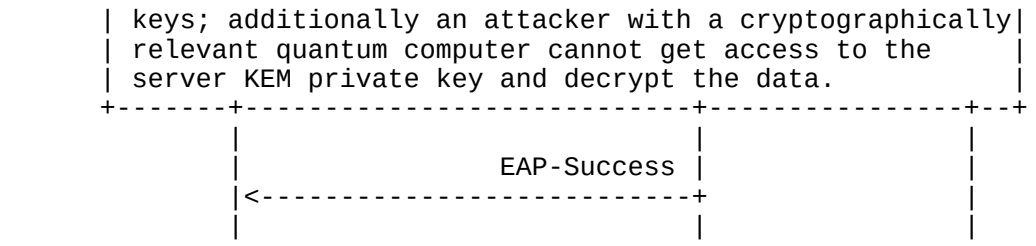
8. Protocol Construction

The above section outlines how the fragmentation works. This section defines the construction for PQC KEM in EAP-AKA’ FS and the other params that are sent via EAP Req/Resp AKA’-Challenge.

8.1. Protocol Call Flow




carries the PQC KEM public key and the AT_KDF_FS
attribute carries PQC KEM algorithm. Both of
these are skippable attributes that can be ignored
if the peer does not support this extension.
EAP-Req/AKA'-Challenge,
AT_RAND, AT_AUTN, AT_KDF,
AT_KDF_FS, AT_KDF_INPUT,
AT_PUB_KEM, AT_MAC
The peer checks if it wants to do the FS extension. If
yes, it will eventually respond with AT_KEM_CT and
AT_MAC. If not, it will ignore AT_PUB_KEM and
AT_KDF_FS and base all calculations on basic EAP-AKA'
attributes, continuing just as in EAP-AKA' per RFC
9048 rules. In any case, the peer needs to query the
auth parameters from the USIM card.
RAND, AUTN
CK, IK, RES
The peer now has everything to respond. If it wants to
participate in the FS extension, it will calculate a
PQC KEM shared secret key based on the server's PQC
KEM public key. Finally, it proceeds to derive all
EAP-AKA' key values and constructs a full response.
EAP-Resp/AKA'-Challenge
AT_RES, AT_KEM_CT,
AT_MAC
The server now has all the necessary values. It
generates the PQC KEM shared secret and checks the RES
and MAC values received in AT_RES and AT_MAC,
respectively. Success requires both to be found
correct. Note that when this document is used,
the keys generated from EAP-AKA' are based on CK, IK,
and PQC KEM shared secret value. Even if there was an
attacker who held the long-term key, only an active
attacker could have determined the generated session



8.2. Key Steps in protocol construction

We outline the following key steps in the protocol:

* Server generates the PQC KEM public key(pk), private key (sk) pair. The server will generate the Authentication Vector (AV). The server PQC KEM key pair is derived as:

  sk, pk = kemKeyGen()

* The server will store the expected response XRES, the PQC KEM private key sk. The server will forward the authenticator part (AUTH) of the AV to peer along with pk.

* The USIM will validate the AUTN received, also verifies the MAC. After the verification is successful and if the peer also supports the Forward secrecy, peer will invoke kemEncaps using pk:

  ct, ss = kemEncaps(pk)

"ct" is the ciphertext from kemEncaps whereas "ss" is shared secret key.

* The peer will send the Authentication result RES and ct to the server.

* The server will verify the RES with XRES. The server will use the ct and PQC KEM private key sk to generate shared secret:

  ss = kemDecaps(ct, sk)

The generated ss from kemDecaps is the shared secret key derived from kemEncaps. The peer and the server first generate the MK_PQ_SHARED_SECRET and subsequently generate MSK, EMSK as shown below:

```
MK = PRF'(IK'|CK',"EAP-AKA'"|Identity)
ct, ss = kemEncaps(pk)
MK_PQ_SHARED_SECRET = PRF'(IK'|CK'|ss,"EAP-AKA' FS"| Identity | ct)
K_encr = MK[0..127]
K_aut = MK[128..383]
K_re = MK_PQ_SHARED_SECRET [0..255]
MSK = MK_PQ_SHARED_SECRET [256..767]
EMSK = MK_PQ_SHARED_SECRET [768..1279]
```

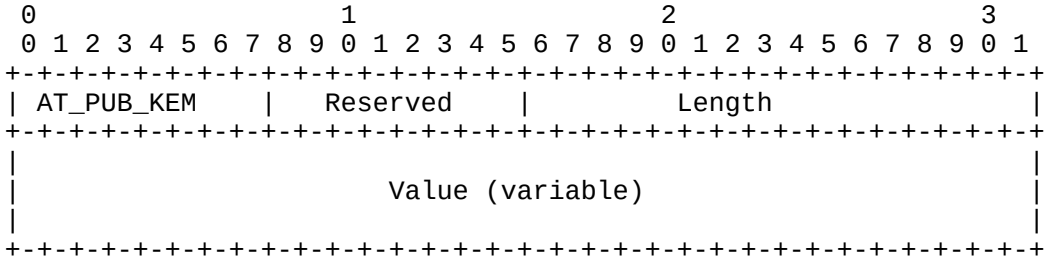
where, pk is PQC KEM public key from the EAP server, ct is the ciphertext from the kemEncaps and it is triggered by the EAP peer only. The pseudo-random function (PRF') binds the shared secret to the ciphertext (ct), achieving MAL-BIND-K-CT.

The ML-KEM already achieves MAL-BIND-K-PK as the hash of the PQC KEM public key is an input to the computation of the shared secret (ss) (line 2 of ML-KEM.Encaps algorithm in [FIPS203]). These computational binding properties for KEMs are defined in [CDM].

9. Extensions to EAP-AKA' FS

9.1. AT_PUB_KEM

The format of the AT_PUB_KEM attribute is shown below.



The fields are as follows:

AT_PUB_KEM:

This is set to TBA1 BY IANA.

Reserved: A 1-byte field reserved for future use. Including this field ensures that the fixed header (Type, Reserved, Length) is 4 bytes long, maintaining 4-byte alignment for the following Value field. The Reserved field MUST be set to 0 on transmission and ignored on receipt.

Length:

A 2-byte unsigned integer indicating the length of this attribute in multiples of 4 octets, including the Type, Reserved, Length, and Value fields, as well as any padding.

The total length of the attribute in octets is obtained by multiplying this field by 4.

This differs from the attribute format used in EAP-AKA [RFC4187], where the Length field is 1 byte.The modification is necessary because PQC KEM public keys, such as those defined in ML-KEM-1024, will be 1568 bytes, which would exceed the 1024-byte limit imposed by the original EAP-AKA format.

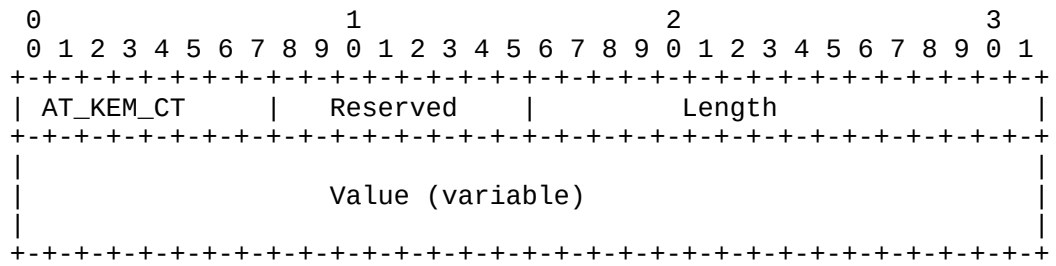
Value:

* EAP-Request: It contains the public key, which is the PQC KEM public key from the EAP server.

Because the length of the attribute must be a multiple of 4 bytes, the sender pads the Value field with zero bytes when necessary. To retain the security of the keys, the sender SHALL generate a fresh value for each run of the protocol.

9.2. AT_KEM_CT

The format of the AT_KEM_CT attribute is shown below.



The fields are as follows:

AT_KEM_CT:

This is set to TBA2 BY IANA.

Reserved: A 1-byte field reserved for future use. The Reserved field MUST be set to 0 on transmission and ignored on receipt.

Length:

A 2-byte unsigned integer indicating the length of this attribute in multiples of 4 octets, including the Type, Reserved, Length, and Value fields, as well as any padding.

The total length of the attribute in octets is obtained by multiplying this field by 4.

This differs from the format used in EAP-AKA [RFC4187], where the Length field is 1 byte. The change is necessary because ciphertexts produced by PQC KEM algorithms,such as 1568 bytes in ML-KEM-1024 will exceed the 1024 byte limit imposed by the original EAP-AKA attribute format.

Value:

* EAP-Response: It contains the ciphertext (ct) from the PQC KEM Encapsulation functio
n from the EAP peer.

Because the length of the attribute must be a multiple of 4 bytes, the sender pads the Value field with zero bytes when necessary. To retain the security of the keys, the sender SHALL generate a fresh value for each run of the protocol.

10. Capability Negotiation

Support for PQC KEM is negotiated using the AT_KDF_FS attribute.

AT_PUB_KEM and AT_KEM_CT use an extended attribute header format that is incompatible with legacy EAP-AKA and EAP-AKA' implementations. Therefore, these attributes MUST NOT be sent unless a mutually supported PQC KEM has been successfully negotiated via AT_KDF_FS.

11. ML-KEM

ML-KEM offers several parameter sets with varying levels of security and performance trade-offs. This document specifies the use of the ML-KEM algorithm at three security levels: ML-KEM-512, ML-KEM-768, and ML-KEM-1024. The main security property for KEMs standardized in the NIST Post-Quantum Cryptography Standardization Project is indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2) (see Section 10.2 of [I-D.ietf-pquip-pqc-engineers]). The public/private key sizes, ciphertext key size, and PQ security levels of ML-KEM are detailed in Section 12 of [I-D.ietf-pquip-pqc-engineers].

12. Security Considerations

The security of the PQ-KEM algorithm depends on a high-quality pseudo-random number generator. For further discussion on random number generation, see [RFC4086].

In general, good cryptographic practice dictates that a given PQ-KEM key pair should be used in only one EAP session. This practice mitigates the risk that compromise of one EAP session will not compromise the security of another EAP session and is essential for maintaining forward security.

Implementations MUST enforce a locally configured maximum Total Attribute Length for fragmented attributes. If the Total Attribute Length exceeds this limit, the attribute MUST be rejected and the authentication exchange aborted. This limit has to be chosen to mitigate DoS attack with support for large PQC key material.

12.1. Selecting ML-KEM Variants

ML-KEM is believed to be IND-CCA2 secure based on multiple analyses. The ML-KEM variant and its underlying components should be selected consistently with the desired security level. For further clarity on the sizes and security levels of ML-KEM variants, please refer to the tables in Sections 12 and 13 of [I-D.ietf-pquip-pqc-engineers].

13. IANA Considerations

Three new Attribute Type values (TBA1, TBA2, and TBA3) from the skippable range are requested from IANA for AT_PUB_KEM (Section 9.1), AT_KEM_CT (Section 9.2), and AT_FRAGMENT (Section 7.1) in the "Attribute Types" registry under the "EAP-SIM/AKA/AKA'" group.

IANA is requested to update the registry "EAP-AKA' AT_KDF_FS Key Derivation Function Values" with the PQC KEM algorithm entries:

Value	Description	Reference
TBA3	EAP-AKA' with MLKEM512	[TBD BY IANA: THIS RFC]
TBA4	EAP-AKA' with MLKEM768	[TBD BY IANA: THIS RFC]
TBA5	EAP-AKA' with MLKEM1024	[TBD BY IANA: THIS RFC]

14. References

14.1. Normative References

[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>.

[RFC2716] Aboba, B. and D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, DOI 10.17487/RFC2716, October 1999, <https://www.rfc-editor.org/rfc/rfc2716>.

[RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <https://www.rfc-editor.org/rfc/rfc4187>.

[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>.

[RFC9048] Arkko, J., Lehtovirta, V., Torvinen, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3GPP Mobile Network Authentication and Key Agreement (EAP-AKA')", RFC 9048, DOI 10.17487/RFC9048, October 2021, <https://www.rfc-editor.org/rfc/rfc9048>.

[RFC9678] Arkko, J., Norrman, K., and J. Preu Mattsson, "Forward Secrecy Extension to the Improved Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' FS)", RFC 9678, DOI 10.17487/RFC9678, March 2025, <https://www.rfc-editor.org/rfc/rfc9678>.

14.2. Informative References

[CDM] "Keeping Up with the KEMs: Stronger Security Notions for KEMs and automated analysis of KEM-based protocols", <https://eprint.iacr.org/2023/1933.pdf>.

[FIPS203] "FIPS-203: Module-Lattice-based Key-Encapsulation Mechanism Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf>.

[FO] "Secure Integration of Asymmetric and Symmetric Encryption Schemes", <https://link.springer.com/article/10.1007/s00145-011-9114-1>.

[HHK] "A Modular Analysis of the Fujisaki-Okamoto Transformation", <https://link.springer.com/chapter/10.1007/978-3-319-70500-2_12>.

[I-D.draft-ar-emu-pqc-eapaka] Banerjee, A. and T. Reddy.K, "Enhancing Security in EAP-AKA' with Hybrid Post-Quantum Cryptography", Work in Progress, Internet-Draft, draft-ar-emu-pqc-eapaka-04, 16 March 2025, <https://datatracker.ietf.org/doc/html/draft-ar-emu-pqc-eapaka-04>.

[I-D.ietf-pquip-pqc-engineers] Banerjee, A., Reddy.K, T., Schoinianakis, D., Hollebeek, T., and M. Ounsworth, "Post-Quantum Cryptography for Engineers", Work in Progress, Internet-Draft, draft-ietf-pquip-pqc-engineers-14, 25 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqc-engineers-14>.

[I-D.ietf-pquip-pqt-hybrid-terminology] D, F., P, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-ietf-pquip-pqt-hybrid-terminology-06, 10 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqt-hybrid-terminology-06>.

[I-D.ietf-tls-hybrid-design] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-16, 7 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-16>.

[NISTFINAL] "NIST Releases First 3 Finalized Post-Quantum Encryption Standards", n.d., <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>.

[PQCAPI] "PQC - API notes", <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/example-files/api-notes.pdf>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

Appendix A. Acknowledgements

This draft leverages text from [RFC9678].

Authors' Addresses

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: kondtir@gmail.com

Aritra Banerjee
Nokia
London
United Kingdom
Email: aritra.banerjee@nokia.com