Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2022

CJ. Tjhai
Post-Quantum
T. Heider
genua GmbH
V. Smyslov
ELVIS-PLUS

July 9, 2021

Beyond 64KB Limit of IKEv2 Payloads
draft-tjhai-ikev2-beyond-64k-limit-01

Abstract

The maximum Internet Key Exchange Version 2 (IKEv2) payload size is limited to 64KB. This makes IKEv2 not usable for conservative post-quantum cryptosystem whose public-key is larger than 64KB. This document discusses the considerations and defines a mechanism to exchange large post-quantum public keys and signatures in IKEv2.

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 January 10, 2022.

Table of Contents

1. Introduction 2
1.1. Terminology 4
2. Proposed Solution Overview 4
3. Protocol Details 6
4. Operational Considerations 8
5. Denial of Service Considerations 8
6. Security Considerations 9
7. IANA Considerations 9
8. References 9
8.1. Normative References 9
8.2. Informative References 10
Appendix A. Alternative Approaches 11
A.1. Hash and URL 11
A.1.1. Key Exchange Payload 11
A.1.2. Certificate Payload 13
A.2. Incremental Transfer and Confirmation 13
Authors' Addresses 14

1. Introduction

Digital communications are secured by public-key cryptography algorithms that rely on computational hardness assumptions such as the difficulty in factoring large integers or that of finding the discrete logarithm on an elliptic curve group or finite-field. Recent advances in quantum computing, however, have caused some concerns on the security of these assumptions. It is conjectured that these hard computational problems can be solved in polynomial time when sufficiently large quantum computers become available. The concerns have prompted the National Institute of Standards and Technology (NIST) to initiate a process to standardize one or more public-key algorithms that are quantum-resistant. This family of algorithms is known as post-quantum or quantum-resistant cryptographic algorithms.

It would be ideal if these cryptographic algorithms can be drop-in replacements to the classical algorithms we currently use. Unfortunately, almost all of the post-quantum cryptography algorithms have either public-key, ciphertext or signature size that is many times larger than their classical counterparts. One of the issues that this will cause, in particular for UDP-based protocols such as IPsec, is fragmentation of packets at IP layer. In the context of IPsec/IKEv2 post-quantum key exchange, the fragmentation issue can be

addressed by sending the post-quantum exchange data in IKE_INTERMEDIATE [I-D.ietf-ipsecme-ikev2-intermediate], which is the intermediary state between IKE_SA_INIT and IKE_AUTH. This is the approach taken in [I-D.ietf-ipsecme-ikev2-multiple-ke] whereby a classical and one or more post-quantum key exchanges are combined in order to establish security associations that are quantum-resistant.

Because all public-key cryptography algorithms rely on computational hardness assumptions, the confidence of a cryptographic algorithm is an important consideration. An algorithm that has been well-studied and field-tested is generally better trusted than newer ones. Unfortunately, the confidence of post-quantum cryptographic algorithms is relatively low. All of the algorithms submitted to NIST post-quantum standardization are based on new computational hardness assumptions and despite being conjectured to be resistant to quantum computer attacks, they have not been well cryptanalyzed compared to the classical counterparts. An exception to this is the Goppa-code based McEliece cryptosystem [McEliece] which has withstood years of cryptanalysis since 1978 and still remains unbroken. It is not surprising that a more efficient and CCA secure version of McEliece cryptosystem, Classic McEliece [CM], is selected as one of the finalists in NIST post-quantum cryptography standardization (at the time of writing this document) [NIST]. Furthermore, this cryptosystem has also been recommended for long-term confidentiality protection of data, see [BSI].

While there is interest in using McEliece cryptosystem, in particular for information that needs to remain secure for a long time, there is a challenge in integrating it with IKEv2 [RFC7296]. One characteristic of McElieces cryptosystem is the very asymmetric size of its ciphertext and public-key. While its ciphertext is the smallest compared to all other post-quantum key-establishment algorithms submitted to NIST, the size of its public-key, however, is the largest. The smallest public-key size of Classic McEliece is 255KB. This presents a problem if one were to use Classic McEliece for key-establishment with IKEv2, as the maximum payload size supported by IKEv2 is limited to 64KB. This document describes a mechanism to support IKEv2 key-exchange with key size larger than 64KB, building on the works in [I-D.ietf-ipsecme-ikev2-multiple-ke] and [I-D.ietf-ipsecme-ikev2-intermediate].

In addition, some post-quantum digital signature algorithms that are finalists or alternate candidates of NIST post-quantum cryptography standardization (at the time of writing this document) [NIST], also have either public key size or signature size greater than 64 KB. This makes impossible to use them in IKEv2 as drop-in replacement for classic signature algorithms.

This document is focused on providing a solution for using large post-quantum algorithms related data (public keys and signatures) in IKEv2. It is not a goal of this document to provide a generic solution to transport large data blocks of arbitrary type in IKEv2.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] and [RFC8174].

This document assumes familiarity with IKEv2 concept described in [RFC7296].

2. Proposed Solution Overview

While the Length field in IKEv2 header has a size of 32 bits, so that the maximum size of an IKEv2 message can theoretically reach 4 GB, the size of any individual payload inside a message is limited to 64 KB due to the fact that the Payload Length field in generic payload header consumes 16 bits only. This makes impossible to transfer blocks of data greater than 64 KB, such as public keys of some post-quantum key exchange methods or some post-quantum signatures. In IKEv2 three types of payloads may contain large amounts of data related to post-quantum algorithms:

- Key Exchange (KE) payload in case of large public key of a post-quantum key exchange method
- Authentication (AUTH) payload in case of large signature of a post-quantum signature algorithm
- Certificate (CERT) payloads in case of large public key of a post-quantum signature algorithm

This specification proposes the following solution to the problem: when block of data of a particular type (public key, signature) exceeds 64 KB in size, it is split into a series of chunks smaller than 64 KB. Each chunk then is placed in its own payload, so that the large block of data is eventually transferred in a series of adjacent payloads of the same type. All these payloads MUST have the same values in their headers (except for Next Payload and Payload Length fields) and MUST be transferred adjacent to each other, so that no other payload should appear between them.

This approach works well for KE and AUTH payloads, since only one such large block is transferred in a message and there is no

ambiguity when it is split over multiple payloads. However, when multiple certificates containing large public keys are transferred and each of them is further splitted into several CERT payloads, there must be a way to find boundaries between these certificates on a receiving side. To solve this problem an empty CERT payload MUST be inserted between other non-empty CERT payloads to mark boundaries between individual certificates. Note that large certificates can also be transferred using "Hash and URL" format (see Section 3.6 of [RFC7296].

The resulting message would exceed 64 KB in size, so that it would not fit into a single UDP datagram. Even if TCP transport [I-D.ietf-ipsecme-rfc8229bis] is used, the size of any individual IKE message in a TCP stream is still limited to 64 KB. For this reason, IKE Fragmentation [RFC7383] MUST be used regardless of the transport protocol if peers are going to transfer large blocks of data. In the case of TCP, the size of fragments is not related to path MTU and can reach 64 KB.

Since IKE Fragmentation is mandatory with this extension and it only can be used on encrypted IKE messages, large blocks of data cannot be transferred in the IKE_SA_INIT exchange.

In encrypted IKE messages, the Encrypted Payload contains other payloads in encrypted form. Since the Payload Length field in the generic IKE payload header has a size of 16 bits, it is impossible to set a proper value for it in the Encrypted Payload header when it contains inner payloads with total length greater than 64 KB. However, since using IKE Fragmentation is mandatory when transferring large blocks of data (even in case of TCP transport), this restriction has no effect. In the case of IKE Fragmentation, the Payload Length field in the Encrypted payload is never transmitted and is used for local processing only. Instead, the IKE message fragments that appear on the wire are limited to 64 KB, so there is no problem with setting a proper value in the Length field of Encrypted Fragment payloads. However, implementations must be prepared that when constructing messages before their fragmentation and after their re-assembly, the total length of the Encrypted payload content may exceed 64 KB.

While mandatory IKE Fragmentation makes it possible to transfer large blocks of data using UDP transport, in practice it may be problematic for the following reason. When fragmenting large messages the number of fragments would be high and all of them are sent at once. If any of these fragment were lost, all the fragments should be re-sent. In congested network environments this would have a negative effect, worsening the congestion. Moreover, the number of IKE message fragments is limited to 2^16. With typical size of IKE message

fragment equal to PMTU or less, this would limit the size of a single large block of data to ~30-100 MB. While this is enough for current applications of this specification, it may be a limitation in the future.

TCP transport has built-in acknowledgement and congestion control mechanisms and does not suffer from these problems. In addition, since the size of IKE message fragments in case of TCP may be up to 64 KB, the size of a single large block of data can in theory reach 4 GB. However, [I-D.ietf-ipsecme-rfc8229bis] implies that if TCP is used as transport for IKE, it is also used for ESP. Encapsulation ESP in TCP has a lot of negative effects on performance and on ESP functionality (see Section 10 of [I-D.ietf-ipsecme-rfc8229bis].

This specifications proposes a mixed transport mode as a solution to the problem. In this mode, IKE uses TCP as its transport, while ESP packets are still sent over IP or are encapsulated in UDP. The use of mixed transport mode is optional and is negotiated in the IKE_SA_INIT exchange.

3. Protocol Details

The initiator starts creating an IKEv2 SA by sending the IKE_SA_INIT request message. If the initiator is going to transfer large blocks of data (e.g. large public keys), then it should make some preparations:

- IKEV2_FRAGMENTATION_SUPPORTED notification MUST be included to negotiate support for IKE Fragmentation

- INTERMEDIATE_EXCHANGE_SUPPORTED notification MUST be included if the initiator proposes key exchange methods with public keys greater than 64 KB

- If the initiator is going to use mixed transport mode then it starts the IKE_SA_INIT request using UDP port 4500 and includes a new status type notification IKE_OVER_TCP (<TBA by IANA>), which has protocol 0, SPI size 0 and contains no data; if the initiator starts the IKE_SA_INIT over TCP, then the mixed transport mode cannot be used and this notification SHOULD NOT be included, it MUST be ignored by the responder if it is still included in the message

Note that UDP port 4500 (and not port 500) is used for the IKE_SA_INIT messages, which is allowed by [RFC7296]. Using port 4500 allows return routability check for UDP messages to be carried out and ensures ESP packets can get through if they are UDP encapsulated.

The responder supporting this specification MUST agree on using IKE Fragmentation by sending back IKEV2_FRAGMENTATION_SUPPORTED notification. If it selects proposal with key exchange method having public key greater than 64 KB, then it MUST agree on using the IKE_INTERMEDIATE exchange by sending back INTERMEDIATE_EXCHANGE_SUPPORTED notification.

If the initiator proposed using mixed transport mode by initiating the IKE_SA_INIT exchange over UDP port 4500 and including IKE_OVER_TCP notification and the responder supports this mode and is willing to use it, then it sends this notification back in the IKE_SA_INIT response. In this case the initiator MUST switch to TCP using destination port 4500 in the next exchange (IKE_INTERMEDIATE or IKE_AUTH) and the responder MUST be prepared to receive the next exchange request message on TCP port 4500. Once switched all subsequent IKE exchanges MUST use TCP transport as described in [I-D.ietf-ipsecme-rfc8229bis], but ESP packets MUST NOT be sent using TCP, instead they are sent either over IP or using UDP encapsulation, depending on the presence of NAT, which is determined in the IKE_SA_INIT exchange.

If the responder does not support mixed transport mode, then it ignores the IKE_OVER_TCP notification and all subsequent IKE exchanges will use UDP transport. Note, that in case the initiator started the IKE_SA_INIT over TCP then the IKE_OVER_TCP notification would not be included in the request message and there would be no option for mixed transport mode.

```
 Initiator                    Responder
 ------------------------------------------------------------------
 HDR, SAi1, KEi1, Ni,
 N(NAT_DETECTION_SOURCE_IP),
 N(NAT_DETECTION_DESTINATION_IP),
 N(IKEV2_FRAGMENTATION_SUPPORTED),
 [N(INTERMEDIATE_EXCHANGE_SUPPORTED),]
 [N(IKE_OVER_TCP)]  --->
                              HDR, SAr1, KEr1, Nr,
                              N(NAT_DETECTION_SOURCE_IP),
                              N(NAT_DETECTION_DESTINATION_IP),
                              N(IKEV2_FRAGMENTATION_SUPPORTED),
                              [N(INTERMEDIATE_EXCHANGE_SUPPORTED),]
                         <---  [N(IKE_OVER_TCP)]
```

Once the IKE_SA_INIT exchange is completed, the peers continue with the following exchanges: one or more IKE_INTERMEDITE exchanges in case multiple key exchanges are negotiated or the IKE_AUTH exchange, as shown below. Note that all messages containing large blocks of

data are sent fragmented using IKE Fragmentation mechanism, but they are not shown here for the sake of simplicity.

```
 Initiator                    Responder
 -------------------------------------------------------------------
 HDR, SK{KEi2.1, KEi2.2, KEi2.3, ...}  --->
                        <---  HDR, SK{KEr2.1, KEr2.2, ...}


 HDR, SK{KEi3.1, KEi3.2, ...}  --->
                        <---  HDR, SK{KEr3.1, KEr3.2, ...}

                        ...

 HDR, SK{IDi, [IDr,] [CERTi1, CERTi2, ...]
 [CERTREQ,] [IDr,] AUTHi1, AUTHi2, ...
 SAi2, TSi, TSr}  --->
                        <---  HDR, SK{IDr, [CERTr1, CERTr2, ...]
                              AUTHr1, AUTHr2, ...
                              SAr2, TSi, TSr}
```

4. Operational Considerations

The IKE fragmentation does not require additional infrastructure, however, there is non-zero probability of lost packets when sending a large number of fragments over a UDP connection. Given a set of fragments, when transmitted, each one of them is not individually acknowledged and if any one of them is lost, the entire set will have to be retransmitted. As a consequence, given the size of the payload and also the potential of multiple retransmissions, there may be a noticeable delay in establishing an security association (SA), in particular in lossy network conditions. Therefore, implementations MAY use a longer timeout value for the purpose of dead-peer detection, but a balance needs to be struck as too large of a value will open up security vulnerabilities as discussed in the following section. In the unlikely event where there is a frequent retransmission due to loss of fragments, implementations MAY send the IKE messages over a TCP connection as specified in [I-D.ietf-ipsecme-rfc8229bis]. If TCP is used as IKE transport, then using mixed transport mode is RECOMMENDED to allow better ESP performance.

5. Denial of Service Considerations

Malicious peers may send a large number of fragments, but incomplete, to the legitimate peer causing memory exhaustion. It is RECOMMENDED that the strategies and recommendations described in [RFC8019] be implemented to counter possible DoS attacks.

An alternative arrangement, if peers do not support [RFC8019], is to allow the transfer of large block of data only after peers are authenticated. In other words, key-establishment using large public-key should not be done to establish an IKE SA, but it should only be used to establish a Child SA or rekeying an IKE SA. In order to protect IKE messages from quantum threats, multiple key-exchanges using a combination of classical and post-quantum ciphers, as described in [I-D.ietf-ipsecme-ikev2-multiple-ke] can be used. Nonetheless, this approach has a limitation whereby if a digital signature scheme with large public-key or signature payload is used, it is still susceptible to DoS attacks.

*** More to be populated in the next version ***

6. Security Considerations

If TCP encapsulation is used, refer to the security considerations in [I-D.ietf-ipsecme-rfc8229bis].

Downloading or transferring large amounts of data is an expensive operation, bandwidth and memory wise. Consequently, implementations should consider using a longer rekeying interval or SHOULD consider relaxing forward secrecy requirements but using CCA-secure key-establishment algorithms only. With chosen-ciphertext attack (CCA)-secure schemes, there is no loss in security if the public-key is reused.

7. IANA Considerations

This document defines a new Notify Message Type in the "Notify Message Types - Status Types" registry:

<TBA> IKE_OVER_TCP

8. References

8.1. Normative References

[I-D.ietf-ipsecme-ikev2-intermediate]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-06 (work in progress), March 2021.

[I-D.ietf-ipsecme-ikev2-multiple-ke]
Tjhai, C., Tomlinson, M., Bartlett, G., Fluhrer, S., Geest, D. V., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in IKEv2", draft-ietf-ipsecme-ikev2-multiple-ke-02 (work in progress), January 2021.

[I-D.ietf-ipsecme-rfc8229bis]
Smyslov, V. and T. Pauly, "TCP Encapsulation of IKE and IPsec Packets", draft-ietf-ipsecme-rfc8229bis-00 (work in progress), April 2021.

[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>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <https://www.rfc-editor.org/info/rfc7383>.

[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>.

8.2. Informative References

[BSI] Federal Office for Information Security, "Cryptographic Mechanisms: Recommendations and Key Lengths", 2020, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf>.

[CM] Classic McEliece submission team, "Classic McEliece: NIST post-quantum cryptography standardization finalist", 2020, <https://classic.mceliece.org/>.

[FIPS-202]
National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", 2015, <https://doi.org/10.6028/NIST.FIPS.202>.

[McEliece]
McEliece, R., "A Public-key Cryptosystem based on Algebraic Coding Theory", DSN Progress Report 42-44, 1978.

[NIST] National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization", <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization>.

[RFC8019] Nir, Y. and V. Smyslov, "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, DOI 10.17487/RFC8019, November 2016, <https://www.rfc-editor.org/info/rfc8019>.

Appendix A. Alternative Approaches

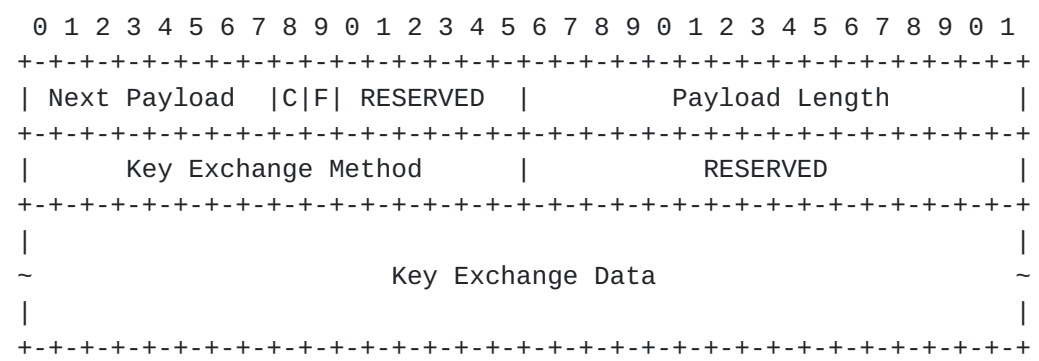
A.1. Hash and URL

[RFC7296] defines a mechanism whereby an authentication payload such as a certificate can be encoded using a hash value and a URL. A peer utilizes HTTP_CERT_LOOKUP_SUPPORTED Notify payload to indicate that X.509 certificates are not transported in-band, instead the other peer shall fetch the certificates from the given URL and verify its integrity from the hash value. In this way, the peer needs to send 20 octets plus a variable length URL only over the wire, instead of a few kilobytes of payload, which is useful in the event IKEv2 message fragmentation is not available.

Large public keys can be transported by reusing the same technique and this can be done in two ways, as described below.

A.1.1. Key Exchange Payload

The Key Exchange Data field of IKEv2 Key Exchange Payload contains a single format, which is a blob that is only meaningful to the specified key exchange method. In order to support hash and URL data, an encoding format needs to be specified on the header.



The reserved bit-field F above specifies the encoding format. If it is 0, the Key Exchange Data is a blob as specified in RFC7296. On the other hand if it is 1, the Key Exchange Data is in the form of hash and URL format, whereby the hash value is the SHA3-256 digest [FIPS-202] of the replaced value truncated to 20 octets and the URL value is a variable length URL (in either http or https schema) that resolves to the DER-encoded of the replaced value itself.

Because the hash and URL value is transported in a Key Exchange Payload, it is possible to support the use-case of a single post-quantum key-establishment with large public-key. This payload will be sent as part of IKE_SA_INIT exchange and it will not require IKE_INTERMEDIATE exchanges.

While using hash and URL method to transport large key-establishment data requires minimal modification to IKEv2 protocol, there are disadvantages from deployment point of view that may make this method impractical. Firstly, an IKE peer that originates a hash and URL value will also need to implement additional infrastructure so that it can serve HTTP requests in order to allow the actual key-establishment data to be fetched. While this may not be an issue for Internet facing peers, in the context of road-warrior or remote-access cases, the hash and URL value is initiated by an IKE peer that is usually a device sitting behind a network address translation (NAT) device and as such, it may not be able to run a publicly reachable HTTP server infrastructure on the same device. An possible solution for these cases is to publish the key-establishment data to a separate server, which is not practical as one cannot expect an IKE initiator to always have deployed an HTTP server. Lastly, IKE peers are predominantly deployed at the network edge where strict firewall rules are generally enforced. The need to open up another port to serve HTTP requests may cause either technical or policy complication that render this approach unacceptable.

The hash and URL approach is vulnerable to (distributed) denial of service attacks as an unauthenticated rogue peer may trick a legitimate peer to fetch a large amount of random meaningless data from a remote server. Implementations SHOULD NOT blindly download all of the data in the given URL. Because a legitimate key-establishment payload should be DER-encoded, they SHOULD download the first few octets to determine the length of the ASN.1 structure representing these octets, then only continue to download the remaining decoded number of octets if the length is expected for the chosen key-establishment algorithm. It should be noted that the content of the data to be downloaded may be under attacker's control and therefore even if the length is as expected, the content may be meaningless bit that is of no use for key-establishment.

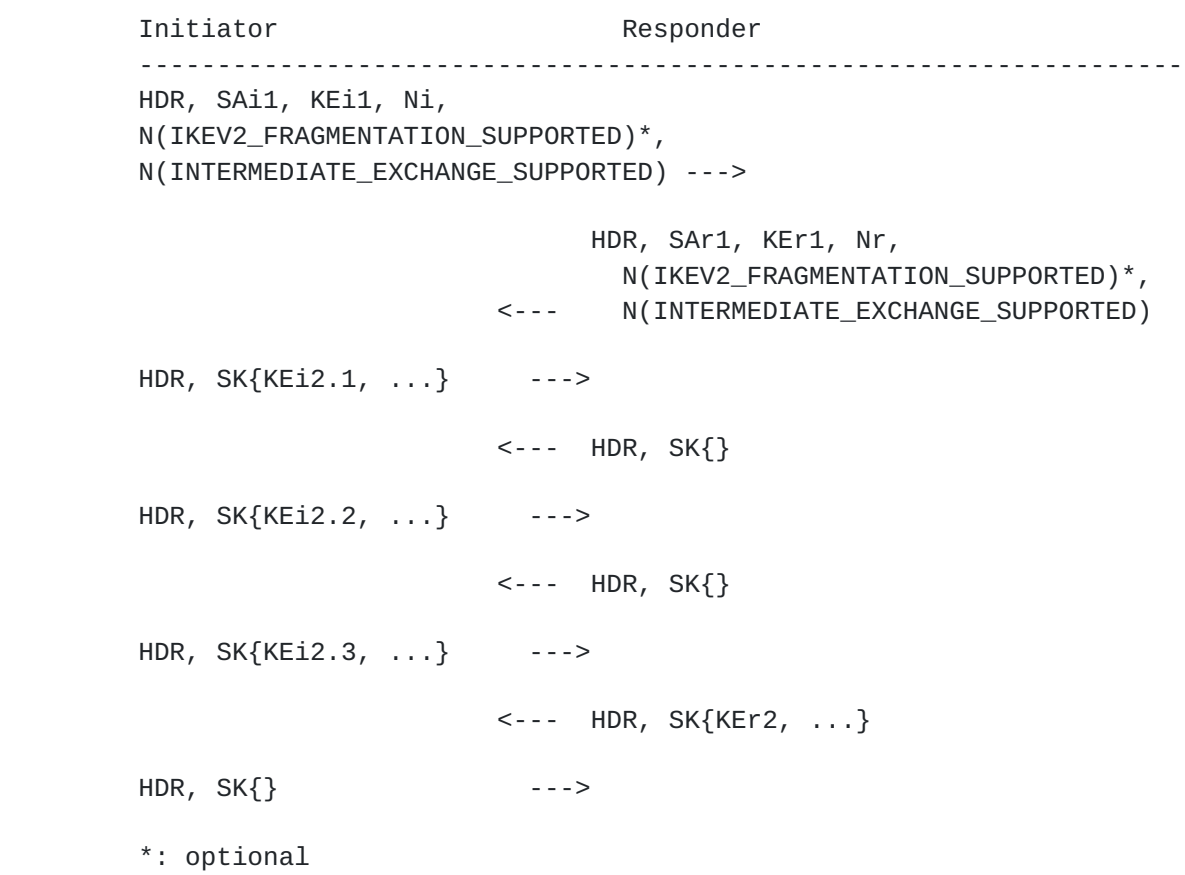
A.1.2. Certificate Payload

An alternative is to re-purpose Certificate Payload to carry the hash and URL value of the post-quantum key-establishment data. At the time of writing, the IANA registry defines two hash and URL encoding values, namely X.509 certificate and X.509 certificate bundle. In order to use this payload, a new encoding value for key establishment data will be required.

Because a Certificate Payload is part of IKE_AUTH message, unlike the previous approach, the hash and URL value of the key-establishment data shall be transported via IKE_INTERMEDIATE message. As such, it will not be able to support a single post-quantum key-establishment with a large public-key case. Furthermore, it is semantically incorrect to re-purpose Certificate Payload, which is intended to carry authentication data, to transport key-establishment data.

A.2. Incremental Transfer and Confirmation

As stated in Section 4 of [RFC7383], if any single fragment is lost, the receiving peer will not be able to reassemble the original large key-establishment payload. The above bulk transfer is susceptible to this issue. There is another way to transfer these payload chunks that is less susceptible to this, but at the cost of higher latency. Instead of transferring in a bulk, each Key Exchange payload chunk must be acknowledged prior to sending the subsequent chunk. As before, the large key-establishment payload is split over several Key Exchange payload chunks where each of them share the same Key Exchange Method value. Each chunk is then sent to the peer using the IKE_INTERMEDIATE message, and each one must be acknowledged by the receiving peer before the subsequent chunk can be sent.



In order to support key-encapsulation mechanism, the receiving peer has to wait until the entire chunks are received before it can respond with its own Key Exchange payload, which may not be large.

Authors' Addresses

CJ Tjhai
Post-Quantum
UK

Email: cjt@post-quantum.com

Tobias Heider
genua GmbH
DE

Email: me@tobhe.de

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru