SPRING Working Group
Internet-Draft: draft-qu-ipv6-quantum-key-sync-negotiation-00
Intended status: Standards Track
Expires: September 11, 2026

J. Qu
PipeChina
L. Chen
PipeChina
J. Zhang
China Academy of Information and Communications Technology
Z. Wei
PipeChina
K. Xu
PipeChina
L. Wang
PipeChina

April 29, 2026

# IPv6 Quantum Key Synchronization and Negotiation

## Abstract

This document provides a mechanism for synchronizing and negotiating quantum keys by carr
ying quantum key information through IPv6 extension headers. The communicating parties se
lect the corresponding encryption and decryption keys from the quantum key library based
on the negotiation results, thereby achieving quantum-encrypted communication over IPv6 n
etwork.



## Status of this Memo

in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents



## 1. Introduction

Traditional encryption typically uses the IPSec method, involving complex IKE protocol interactions. IPSec primarily employs the IKE (Internet Key Exchange) protocol to negotiate keys and SAs (Security Associations). IKE is divided into two phases. In Phase 1 of IKE, the two communicating



parties exchange identity information and algorithms supported with each other, use Diffie-Hellman (DH) for key exchange to generate a shared key, and establish a bidirectional IKE SA security channel, which includes encryption algorithms, verification algorithms, lifetimes, etc., these will be provided to subsequent IKE Phase 2 negotiations. In IKE Phase 2, based on the IKE SA established, IPSec parameters are negotiated, such as ESP/AH, encryption/verification algorithms, lifetimes, etc. The key material generated in Phase 1 is used to derive the IPSec SA key, which is used for data protection.

Once the data stream to be protected by encryption is determined, each party in communication establishes a pair of inbound and outbound IPSec SAs for data encryption and decryption. The data is encapsulated using IPSec SA (either ESP or AH) and transmitted in the transport mode or tunnel mode configured. To ensure security, IPSEC SAs need to be updated regularly based on the time or number of bytes of encrypted data set.

According to the principles of IKE, in the first phase of IKE negotiation, a symmetric key shared between the two communicating parties is generated using the Diffie-Hellman public-key mechanism. During the calculation of the symmetric key, the two communicating parties pre-agree on two public parameters: a large prime number p and a primitive root (generator) g from modulo-p. Each party uses their private confidential random integer a and b respectively to compute public values A and B, and sends A and B to the other party through the network. The two communicating parties then calculate the shared key K based on the received A/B from each other. Based on this principle, the private values a, b and the shared key K are never transmitted through the network. Therefore, in non-quantu

m computing scenarios, even if an attacker intercepts p, g, A, and B, it is difficult to reverse-engineer a, b, or K from this information because it requires solving the discrete logarithm problem, which is hard to achieve with traditional computing but can be easily solved with quantum computing. Thus, in IPv6 applications, it is necessary to address the insecurity of the DH algorithm in generating symmetric keys and also to provide encryption communication capabilities for future IoT and other lightweight protocol terminals. This is where quantum encryption computing comes into play.

### 1.1. Requirements Language

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

## 2. IPv6 Quantum Encryption Process




In the scenario of IPv6 data quantum encryption transmission, Quantum Key Distribution (QKD) networks can be used to generate symmetric keys. The IPv6 transmission network adopts the IPv6 extension header to lightly synchronize and negotiate keys between the two parties of

encrypted communication, thereby completing secure and reliable encrypted communication.

IPv6 transmission network and Quantum Key Distribution (QKD) network work in collaboration. At network terminal devices A and B, corresponding quantum key server (QKS) are respectively present to apply for and distribute keys. Network terminal devices and quantum key servers (QKSs) can have a one-to-one relationship or a many-to-one relationship, as shown in Figure 1 below.

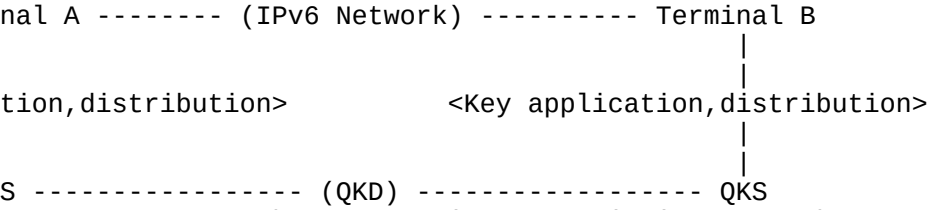



Figure 1 IPv6 quantum key encryption transmission network

## 3. IPv6 Key Acquisition and Management

An important prerequisite for IPv6 communication devices to conduct quantum-encrypted communication is to obtain quantum keys from the QKD network and effectively manage the obtained keys. After IPv6 communication terminals acquire quantum keys, they perform classification management and lifecycle management of the quantum keys locally to meet the key requirements for different authentication and encryption methods during communication.

### 3.1. IPv6 Quantum Key Acquisition

The password application system APP on terminal confirms the user’s identity with the authentication and authorization center and applies for an identity credential from the quantum key server. Then, the APP obtains the quantum key based on the identity credential. The APPs on the sender and receiver can obtain quantum keys corresponding to the key topic (Topic), sender QKS(Quantum Key Server) identifier, URI of the sender APP, receiver QKS identifier, and URI of the receiver APP. Keys can be obtained online or offline.

The IPv6 network terminal and QKD network use multiple methods to ensure security when transmitting keys. First, the quantum key distribution system

remains physically isolated from the user’s business system. Second, when the encryption application system (APP) connects to the quantum key server (QKS), HTTPS (Web-style interface) or TLS/SSL (binary-style interface) is used to provide confidentiality and integrity protection for data transmitted between them. Third, when the APP is limited by its own storage, computing resources, etc., and cannot use HTTPS or SSL to communicate securely with the quantum key service system, key data can be protected at the application layer. That is, when the APP and the quantum key server transmit key data (quantum keys or user keys), confidentiality and integrity protection of the key data are provided.

The terminal device obtaining quantum keys from the Quantum Key Distribution (QKD) network, as well as the security guarantees for obtaining quantum keys, are outside the scope of this document.

### 3.2. IPv6 Quantum Key Management

After the APP on the IPv6 communication terminal obtains quantum keys from QKD, it is necessary to manage the keys in a reasonable and effective manner. At the same time, to meet the requirements of authentication and encryption, each quantum key also needs to derive authentication keys and quantum keys.

#### 3.2.1. Key Pool Structure and Storage

The quantum key pool is maintained by IPv6 terminal devices (or gateway devices), and each original quantum key record contains:

Table 1 Original Quantum Key Pool

| Index | KEYID | Original Key | Key Status |
|---|---|---|---|

Index — Key index number, which is the sequential index of the key stored in the key pool. It takes natural numbers 1, 2, 3, etc., in order. A natural number sequence is obtained by calculating the KEYID in the key pool, and the keys are arranged in ascending order of their natural number sequence and stored sequentially in the key pool. The Index is the sequential number of this ordered storage.

KEYID — Unique key identifier, with a length of 8-16 bytes. The KEYID is derived from the Quantum Key Ticket (QKT), which is provided by the QKD system. The QKT contains unique attributes of the quantum key (such as generation time, channel ID, key length), ensuring that the KEYID is globally unique and unpredictable, thereby avoiding the risk of key identifier leakage.

Original Key — The original key generated by QKD, with a length of 16-32 bytes.

Key Status — Idle (unused), Pending (negotiating), Active (bound and in use).


Internet-Draft IPv6 Quantum Key Negotiation April 2026

Two subkeys are derived from each original quantum key (corresponding to KEYID):

Authentication Key (AuthKey): Used for integrity verification of the AH extension header (e.g., HMAC-SHA256).

Encryption Key (EncKey): Used for symmetric encryption of the ESP extension header (e.g., AES-256-GCM).

Terminal devices or gateway devices maintain an association table. Through the KEYID of the original key, the derived AuthKey and EncKey can be directly queried, as shown in the following table:

Table 2 Key Association Relationship

| KEYID | Original Key | AuthKey | EncKey |
|---|---|---|---|

KEYID 8-16 bytes. Unique Identifier for the Key generated based on QKT.

Original Key 16-32 bytes. Original Key generated by QKD.

AuthKey 16-32 bytes. Derived Authentication Key.

EncKey 16-32 bytes. Derived Encryption Key.

All keys, including original keys, authentication keys, and encryption keys, are stored in an encrypted form.

The specific derivation method of the keys is outside the scope of this document.

3.2.2. Key Status Management

(1) The Key Status is Defined as Follows:

a. Original Key Status:



Idle: Not assigned or not under negotiation, can be selected for use.

Pending:Negotiation has been initiated but not confirmed.

Active: Already bound to a communication session (such as IPsec SA).

b. Derived Key Status:

Idle: Derivation is complete but not bound for use.

Active: Already bound to a specific protocol (such as AH or ESP).

Expired:Exceeded lifecycle or destroyed.

(2) Key Status Transition Rule:

The key state transition rules when using only the encryption key are as follows in the table below.

Table 3 Key Status Transition Rule 1

| Event | Original Key Status | Derived Authentication Key Status | Derived Encryption Key Status |
|---|---|---|---|
| Key Generated | Idle (Initial) | Idle (Derived) | Idle (Derived) |
| Key Negotiation, for Encryption | Pending | Idle (Derived) | Idle (Derived) |
| Negotiation Confirmed | Active | Idle (Derived) | Active (Protocol Bound) |
| Used up or Expired | Expired (Destroy) | Expired (Destroy) | Expired (Destroy) |

The key state transition rules when using both authentication and encryption keys are as follows in the table below.



Table 4 Key Status Transition Rule 2

| Event | Original Key Status | Derived Authentication Key Status | Derived Encryption Key Status |
|---|---|---|---|
| Key Generated | Idle (Initial) | Idle (Derived) | Idle (Derived) |
| Key Negotiation, Authentication and Encryption | Pending | Idle (Derived) | Idle (Derived) |
| Negotiation Confirmed | Active | Active (Protocol Bound) | Active (Protocol Bound) |
| Used up or Expired | Expired (Destroy) | Expired (Destroy) | Expired (Destroy) |

(3) Key Status Update Mechanism:

After each negotiation or destruction operation, the terminal device or g

ateway device updates the status field in the key associated relationship table. The rece iver synchronously updates the local key pool status after confirming the derived key sta tus.

For example, when an IPv6 terminal initiates a key negotiation, the origi nal key status changes from Idle to Pending, and the derived key status is Idle. After th e negotiation is confirmed, the original key status changes to Active, and the derived ke y status changes to Active, indicating that the derived key has been bound to the IPv6 en crypted communication session.To enhance the security of encrypted communication, a one-t ime pad is typically used. So once the key is used up or expires, the original key and de rived keys are destroyed.

# 4. IPv6 Key Synchronization Process

## 4.1. IPv6 Key Synchronization Process

In quantum encryption systems, the two communicating parties in an IPv6 network p erform encrypted communication after they obtain keys from the
Quantum Key Distribution (QKD) network. This process relies on the strict consist ency of the key status between the two communicating parties. Although the QKD network ca n ensure that the keys sent to both parties are synchronized, the two parties still need to perform a synchronization before communication. Especially in scenarios of multi-point mutual communication, each communicating party can only choose an idle key to



initiate a key negotiation and proceed with the next step of encrypted transmissi on by synchronizing their respective key usage situations. Without the synchronization st ep, the following issues may arise:

Inconsistent key status: Quantum keys generated by QKD devices typically have a l ifecycle (e.g., limited by a time window or usage count). If one party has already consum ed the key while the other still considers it available, this will lead to encryption fai lure or the risk of key reuse.

Key availability verification: QKD keys may become invalid due to channel noise, failed eavesdropping detection, or storage anomalies. The synchronization process can con firm the availability of keys, preventing the use of unreliable keys.

Resistance to quantum computing attacks: The unconditional security of QKD keys r equires them to be used for only one encryption. Synchronization ensures both parties str ictly adhere to the ’one-time pad’ principle, preventing key reuse.

After the IPv6 communication terminal has obtained the key resource pool from QKD , both communicating parties need to synchronize the status of keys in each other’s local key pools. Both parties need to synchronize the three states of each key:
Idle(value 00): Generated but not activated
Pending(value 01): The key is being negotiated but not yet ready
Active(value 10): The key is currently being used for encrypted communication

Both communicating parties need to generate a key status comparison table. In a n ewly initiated communication process, only keys where both parties are in the Idle state can be used. Assuming the communicating parties are A and B, each has 10 keys obtained fr om QKD in their respective key pools. A synchronization table is generated through status synchronization as follows:

Table 5 Key Pool Status Synchronization

| Key Index | Key Status on Device A | Key Status on Device B |
|---|---|---|
| 1 | Idle (00) | Pending(01) |
| 2 | Active (00) | Idle (00) |
| 3 | Pending(01) | Idle (00) |

| 4-10 | Idle (00) | Idle (00) |
|---|---|---|



According to the status in the key synchronization table, the first three keys status between the two communicating parties are inconsistent. There are seven keys available for use in this encryption communication, which can be selected from numbers 4 to 10. When negotiating key in the next phase, key number 4 can be prioritized as the negotiation key in sequence.

The quantum key synchronization process between the two communication parties A and B is as follows:

participant A as Initiator

participant B as Responder

A->>B: REQ_SYNC (A sends its key status to B by carrying a status bitmap of keys in the local key pool)

B->>A: ACK_AVAILABLE (B sends its own key status bitmap to A after confirming the key status of A has been received)

### 4.2. IPv6 Key Synchronization Packet Encapsulation

The synchronization of key status is performed at both ends of IPv6 encrypted communication. Key status information only needs to be processed by the destination node of the packet, so IPv6 DOH (Destination Options Header) is used to carry key status information. If the packet is encapsulated with HBH (Hop-by-Hop Options Header) or RH (Routing Header), this DOH is encapsulated after them. According to the definition in RFC 8200, The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header and has the following format:

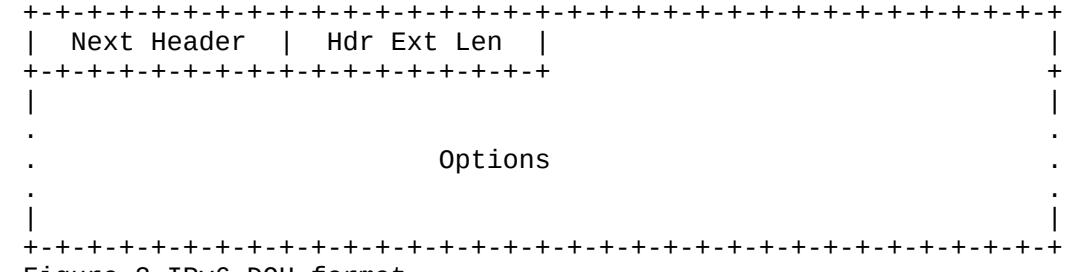



Figure 2 IPv6 DOH format

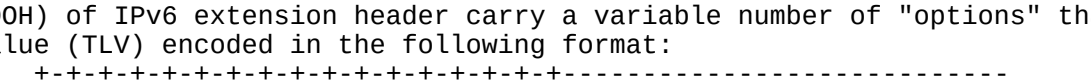
Next Header — 8-bit selector. Identifies the type of header immediately following the Destination Options header.

Hdr Ext Len — 8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.



Options — Variable-length field, of length such that the complete Destination Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options.

According to the definition in RFC 8200, the options in Destination Options header (DOH) of IPv6 extension header carry a variable number of "options" that are type-length-value (TLV) encoded in the following format:



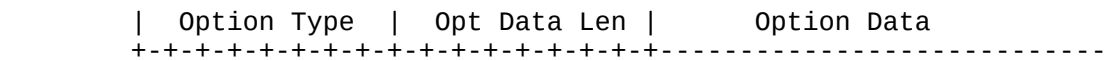



Figure 3 IPv6 DOH Options TLV format

Option Type — 8-bit identifier of the type of option. Represents the quantum key message option, recommended value 0x50 (binary 0101 0000). The highest two bits ’01’ indicate ’discard the packet if the processing IPv6 node does not recognize the Option Type’, and third-highest-order bit of the Option Type ’0’ indicates ’the Option Data does not change en route’.

Opt Data Len — 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

Option Data — Variable-length field. Option-Type-specific data.

Option Data is used to implement key status synchronization. The encapsulation format of REQ_SYNC and ACK_AVAILABLE messages is as follows:

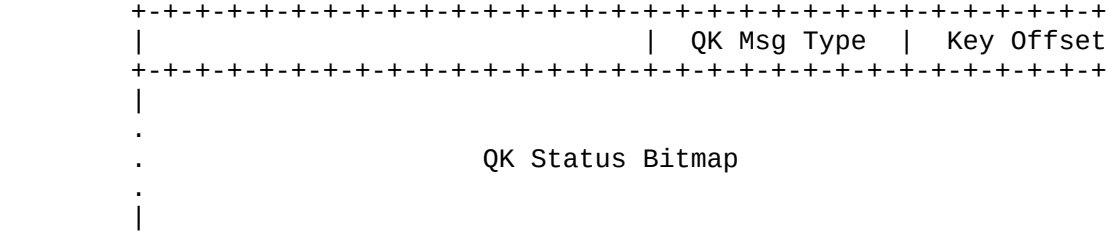



Figure 4 IPv6 Quantum Key Synchronization Message Encapsulation

QK Msg Type — 8-bit identifier of the type of QK message.Message type specifying the quantum key, with defined format:

0x00: Quantum Key Status Synchronization Request
0x01: Quantum Key Status Synchronization Response



0x02: Quantum Key Negotiation Request
0x03: Quantum Key Negotiation Confirmation
0x04-0xFF: Reserved

Key Offset — Key offset, with a value range of 0-255. This is used to specify which batch of keys in the key pool the synchronized key belongs to. When there are a large number of keys, it is impossible to send the status of all keys in a single message. Therefore, keys need to be sent in batches, and this key offset is carried in the synchronization message. According to the key generation speed and the frequency of quantum key replacement every 10 seconds, the key pool of both communication parties usually does not exceed 1 million keys. Each key requires 2 bits to represent its status, so 100,000 keys require 200,000 bits (25KB) to represent their statuses. Using DEFLATE compression can reduce 25KB to 3,000-5,000 bytes, which meets the MTU requirements of an IPv6 packet. Therefore, each synchronization message can send the status of 100,000 keys for grouped synchronization. When the Offset in the synchronization message is to N, the batch of keys sent this time starts from 10*N + 1.

QK Status Bitmap Quantum Key Status Bitmap. The REQ_SYNC message carries the requester’s key status bitmap, and the ACK_AVAILABLE message carries the responder’s key status bitmap.

The total length of the bitmap is determined by the number of keys in the key pool. Each key requires 2 bits to identify its status, so n keys need a 2*n bit length to represent their statuses. Taking 10 keys as an example, the bitmap length for their key statuses is 20 bits (10 keys * 2 bits per key). To maintain overall 8-byte alignment of the DOH header, 4 bits need to be padded and set to 0, resulting in a total bitmap length of 3 bytes (24 bits). The keys are arranged from high to low bit positions, corresponding sequentially to KEY1 through KEY10, with each group of 2 bits representing the status of one key:

00:Idle
01:Pending
10:Active
11:Invalid or blank key



Table 6 Example of Key Pool Status Identifier

| Index | KEY ID | Status on Device A | Status on Device B |
|---|---|---|---|
| 1 | KEYID-A | Pending(01) | Idle(00) |
| 2 | KEYID-C | Idle(00) | Pending(01) |
| 3 | KEYID-E | Active(10) | Active(10) |
| 4 | KEYID-G | Pending(01) | Pending(01) |
| 5 | KEYID-F | Idle(00) | Idle(00) |
| 6 | KEYID-D | Idle(00) | Idle(00) |
| 7 | KEYID-Q | Idle(00) | Idle(00) |
| 8 | KEYID-P | Idle(00) | Idle(00) |
| 9 | KEYID-X | Idle(00) | Idle(00) |
| 10 | KEYID-Y | Idle(00) | Idle(00) |

So the bitmap of 10 keys, from high bit to low bit, corresponds to 1 to 10 of index, which is 01001001 00000000 0000. After padding to the required
number of bits, it becomes 01001001 00000000 00000000, which converts to hexadecimal as 0x49 0x00 0x00.

## 5. IPv6 Key Negotiation Process

### 5.1. IPv6 Key Negotiation Process

Key synchronization addresses the problem of sharing key status. To further determine the specific keys used by both communicating parties, key negotiation is still required, with its core purpose is:

Key selection and binding: Both communicating parties may obtain multiple keys from QKD devices (e.g., keys generated in different time windows). The negotiation process

binds quantum keys with security parameters (such as SPI, protocol configuration) in IPv6 extension headers (e.g., AH/ESP) to form executable encryption rules.

Secure Session establishment: The negotiation process needs to bind the quantum key with security parameters (SPI, protocol configuration) in IPv6 extension headers (such as AH/ESP), forming executable encryption rules.




Prevention of man-in-the-middle interference: Key negotiation transmits key identifiers (such as quantum key tickets QKT) through a secure channel (e.g., an encrypted classical signaling channel), ensuring that the key is not tampered with or replaced by third parties during use.

After the two parties in IPv6 encrypted communication complete quantum key synchronization and share a consistent key status table, specific encryption key negotiation is performed through the IPv6 packet extension header DOH. The negotiation needs to complete the following steps:

1. Key Selection: The communication sender dynamically selects a quantum key that both parties are in the Idle state.

2. Key Binding: Bind the selected quantum key with the current communication session.

The quantum key negotiation process between A and B is as follows:
a. Participant A as the sender
b. Participant B as the receiver
c. Sender A --> Receiver B: REQ_NEGOTIATE (carrying the INDEX of the key that is idle for both A and B)
d. alt KEYID is valid
e. Receiver B --> Sender A: CONFIRM_NEGOTIATE (success)
f. else KEYID is invalid
g. Receiver B --> Sender A: CONFIRM_NEGOTIATE (failure + error code)
h. end

Assuming after quantum key synchronization between A and B, the quantum key synchronization status tables maintained by A and B are as follows. The sender A will select the key with ID 4 to initiate the key negotiation process with receiver B.

Table 7 Example of Quantum Key Synchronization Status

| Key Index | Key Status on Device A | Key Status on Device |
|---|---|---|
| 1 | Idle (00) | Pending(01) |
| 2 | Active (00) | Idle (00) |
| 3 | Pending(01) | Idle (00) |
| 4-10 | Idle (00) | Idle (00) |

5.2. IPv6 Key Negotiation Packet Encapsulation




(1) REQ_NEGOTIATE Message

Key negotiation is performed at both ends of IPv6 encrypted communication. Key negotiation messages only need to be processed by the destination node of the packet, so IPv6 DOH (Destination Options Header) can be used to carry the negotiated key information. Additionally, if the packet is encapsulated with HBH (Hop-by-Hop Options Header)

or RH (Routing Header), this DOH is encapsulated after them.

According to the definition in RFC 8200, the options in Destination Options header (DOH) of IPv6 extension header carry a variable number of "options" that are type-length-value (TLV) encoded in the following format:

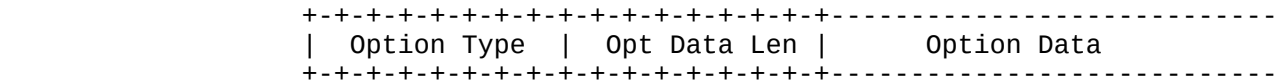



Figure 5 IPv6 DOH Options TLV format

| | |
|---|---|
| Option Type | 8-bit identifier of the type of option. Represents the quantum key message option, recommended value 0x50 (binary 0101 0000). The highest two bits ’01’ indicate ’discard the packet if the processing IPv6 node does not recognize the Option Type’, and third-highest-order bit of the Option Type ’0’ indicates ’the Option Data does not change en route’. |
| Opt Data Len | 8-bit unsigned integer. Length of the Option Data field of this option, in octets. |
| Option Data | Variable-length field. Option-Type-specific data. |

Option data is used to implement key negotiation. The encapsulation format of the REQ_NEGOTIATE message is as follows:

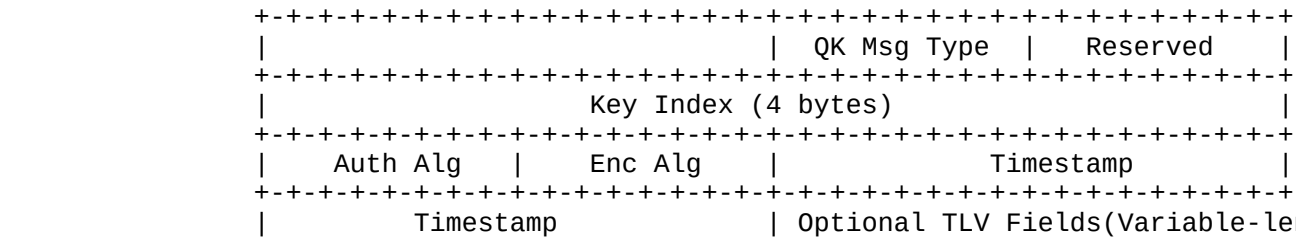



Figure 6 IPv6 Quantum Key negotiation request Message Encapsulation

| | |
|---|---|
| QK Msg Type | 8-bit identifier of the type of QK message. Message type specifying the quantum key, with defined format:<br>0x00: Quantum Key Status Synchronization Request<br>0x01: Quantum Key Status Synchronization Response<br>0x02: Quantum Key Negotiation Request<br>0x03: Quantum Key Negotiation Confirmation<br>0x04-0xFF: Reserved |
| Reserved | 8-bit reserved field. The default value is zero, used for future alignment expansion. |
| Key Index | 4-byte key index value. A globally unique identifier used for quantum key authentication. |
| Auth Alg | 8-bit unsigned integer. Represents the authentication algorithm, and the algorithm value is as follows: |

0x01:Quantum Authentication Protocol
0x02:HMAC-SHA256
0x03:OTP based on quantum random numbers

Enc Alg 8-bit unsigned integer. Represents the encryption algorithm, and the algorithm value is as follows:

0x01:AES-256-GCM
0x02:Quantum-safe algorithm Kyber-1024

Timestamp 4-byte UTC timestamp or monotonically increasing sequence number, used for anti-replay protection.

Optional TLV Fields Variable-length optional TLV field, using the standard Type-Length-Value structure, which supports backward compatibility:

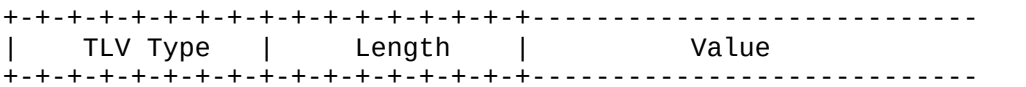



Figure 7 Type-Length-Value Structure

Future supported TLV types that can be expanded to include:

0x01:Key Survival Time (4 bytes integer, unit: seconds)

0x02:Quantum random number seed (variable length)

0x03:Quantum Channel ID (8 bytes)



(2) CONFIRM_NEGOTIATE Message

The encapsulation format of CONFIRM_NEGOTIATE packet is as follows:

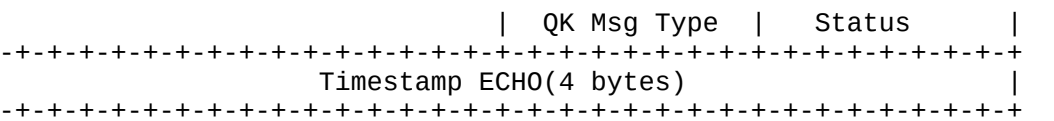



Figure 8 IPv6 Quantum Key negotiation confirm Message Encapsulation

QK Msg Type 8-bit, identifier of the type of QK message. Message type specifying the quantum key, with defined format:
0x00: Quantum Key Status Synchronization Request
0x01: Quantum Key Status Synchronization Response
0x02: Quantum Key Negotiation Request
0x03: Quantum Key Negotiation Confirmation
0x04-0xFF: Reserved

Status 8-bit unsigned integer, It represents the. specific quantum key negotiation result, with the following values:
0x00: Success (Key has been bound)
0x01: Invalid KEYID (QKT does not exist)
0x02: The key status is not Idle
0x03: HMAC verification fail
0x04: Timestamp expired (exceeded time window)

| | |
|---|---|
| Timestamp ECHO | 4-byte. Return the timestamp in the REQ_NEGOTIATE message as is, for anti-replay association. |

After the IPv6 communication terminal obtains a quantum key from QKD, an authentication key and an encryption key are derived based on the key using the same algorithm. Therefore, during key negotiation, only the original quantum key KEYID needs to be negotiated. Both parties find the corresponding authentication and encryption keys based on the original quantum key KEYID to perform encrypted communication. The specific authentication and encryption encapsulation processing procedures are not within the scope of this paper.

## 6. IANA Considerations

In the IPv6 Destination Options Header (DOH), the allocation of Option Type follows the unified management of IANA (Internet Assigned Numbers Authority). The following is a comprehensive description of well-known used Option Types and the available range for applications.




Table 8 Option Types well-known Used in DOH

| Option Type (Hexadecimal) | Name | Purpose Description | Source Standard |
|---|---|---|---|
| 0x00 | Pad1 | Fill 1-byte gap, Purpose Description | RFC 8200 |
| 0x01 | PadN | Fill N-byte (The Length field specifies the length) | RFC 8200 |
| 0x04 | Tunnel Encapsulation | Tunnel encapsulation information(such as IPv6-in-IPv6) | RFC 2473 |
| 0x05 | Router Alert | Notify the router that special handling is required(e.g.,RSVP) | RFC 2711 |
| 0x08 | CALIPS0 | Network architecture with support for multi-level security labels | RFC 5570 |
| 0xC9 | Home Address | Home Address used to identify mobile nodes in Mobile IPv6 | RFC 6275 |
| 0x8B | Jumbo | Support for ultra-large payloads exceeding 65535 bytes (requires hop-by-hop option headers) | RFC 2675 |

Note: Some Option types (such as Pad1/PadN) are shared basic types used by both DOH and the Hop-by-Hop Options Header (HBH).

The message for synchronizing and negotiating quantum keys in IPv6 network requires an Option Type. According to the functional requirements of the message, the highest three bits are set to 010, so 0x50 is applied to IANA as the Option Type.

## 7. Security Considerations

IPv6 quantum key communication itself is an important means of data security communication. The technical solution described in this document can be regarded as an enhancement of the security and reliability of data communication based on quantum encryption technology.

The IPv6 network terminal and the QKD network use multiple methods to ensure security when transmitting keys, as described in the IPv6 quantum key acquisition.

# 8. References

## 8.1. Normative References

[RFC8200] S. Deering, R. Hinden " Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, < https://www.rfc-editor.org/info/rfc8200>.

## 8.2. Informative References

TBD

# 9. Acknowledgments

The authors gratefully acknowledge the many helpful suggestions of the members of the XXX Working Group, the End-to-End Protocols research group, and the Internet community at large.

The authors would like to thank the following for their valuable contributions of this document:

# Authors’ Addresses

Jiewu Qu
PipeChina

Email: qujw@pipechina.com.cn

Liang Chen
PipeChina

Email: chenliang14@pipechina.com.cn

Jiandong Zhang
China Academy of Information and Communications Technology

Email: zhangjiandong@caict.ac.cn

Zheng Wei
PipeChina

Email: weizheng@pipechina.com.cn

Kun Xu
PipeChina

Email: xukun@pipechina.com.cn



Hao Tang
China Academy of Information and Communications Technology

Email: tanghao@caict.ac.cn

Li Wang
PipeChina

Email: wangli@pipechina.com.cn

Ziqing Wang
PipeChina

Email: wangzq30@pipechina.com.cn

Yuchen Jiang
PipeChina

Email: jiangyc02@pipechina.com.cn

Peichun Yuan
PipeChina

Email: yuanpc@pipechina.com.cn

Yuehao Guo
PipeChina

Email: guoyh09@pipechina.com.cn

Qiushi Feng
PipeChina

Email: fengqs01@pipechina.com.cn