Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 7 March 2026

R. Singh, Ed.
C. Hill
Cisco Systems, Inc.
S. Kawaguchi
J. Lupo
QuSecure, Inc.

3 September 2025

Secure Key Integration Protocol (SKIP)
draft-cisco-skip-02

Abstract

This document specifies the Secure Key Integration Protocol (SKIP), a two-party protocol that allows a client to securely obtain a key from an independent Key Provider. SKIP enables network and security operators to provide quantum-resistant keys suitable for use with quantum-resistant cryptographic algorithms such as AES-256. It can also be used to provide an additional layer of security to an already quantum-resistant secure channel protocol for a defense-in-depth strategy, and/or enforce key management policies.

Table of Contents

1. Introduction

Many existing secure channel protocols such as the Internet Key Exchange Protocol Version 2 (IKEv2) and Transport Layer Security (TLS) utilize public key cryptography that is vulnerable to Cryptographically Relevant Quantum Computers (CRQC)[PQCRYPTO]. One solution to mitigate this threat is to replace the vulnerable public key algorithms with different algorithms that are believed to be resistant to quantum computers. An alternate solution is to augment the original protocol by providing each protocol principal with a pre-shared key. If the pre-shared key has sufficient entropy and is mixed into the protocol's key derivation process in a quantum-resistant manner (such as via a pseudorandom function (PRF) instantiated using symmetric cryptography primitives), and other encryption and authentication algorithms are quantum-resistant, then the whole system is considered to be quantum-resistant. Many secure channel protocols already support the ability to mix a pre-shared key into their key derivation process. For example, [RFC8784] specifies an IKEv2 extension that utilizes a post-quantum pre-shared key (PPK) in order to provide quantum resistance to the IKEv2 handshake and the resulting IPsec session.

One common solution to distributing these pre-shared keys is to use out of band mechanism. This approach, however, has a number of drawbacks. For one, the administrative burden of installing the keys scales quadratically with the number of peers. Second, key management best practices suggests periodic rotation of keys, which requires additional and recurring support. Lastly, manual administration increases the likelihood that a low entropy key (e.g,. a password) is chosen. This misconfiguration would degrade any quantum resistance security benefits that we hope to achieve by mixing in the key in the first place. Instead, a more dynamic and automated source of key provisioning should be preferred [RFC4107].

This document describes the Secure Key Integration Protocol (SKIP), a protocol designed to facilitate the dynamic provisioning of keys to protocol principals. SKIP operates in a model where the two principals, called encryptors, are situated at each end of a point-to-point connection. Each encryptor is associated and co-located with a Key Provider (KP). Each KP is capable of producing the same key upon request from its associated encryptor, allowing this key to function as a pre-shared key between the two encryptors. For example, when integrated with the IKEv2 PPK extension (refer Section 5), SKIP can be used to provide each peer with a fresh PPK per session. Furthermore, SKIP defines a method by which a KP can provide entropy to an encryptor.

SKIP defines a modular and extensible security architecture. The keys provided to encryptors can be used to provide quantum-resistance to vulnerable secure channel protocols without reducing security guarantees against classical (i.e., non-quantum) adversaries, provide an additional layer of security to an already quantum-resistant secure channel protocol for a defense-in-depth strategy, and/or enforce key management policies. It imposes no restriction on the means by which two KPs synchronize a key. KPs may employ one or more technologies believed to be quantum-resistant, including, but not limited to post-quantum cryptography (PQC), quantum key distribution (QKD), a trusted third-party protocol, or a one-time pad (OTP) [CSFC]. The KPs can be upgraded, replaced, or reconfigured independent of the underlying encryptors, supporting goals such as cryptographic agility, defense-in-depth, and high availability. Moreover, SKIP can help enforce key management and rotation policies for any protocol that supports the use of a pre-shared key, such as Media Access Control Security (MACsec).

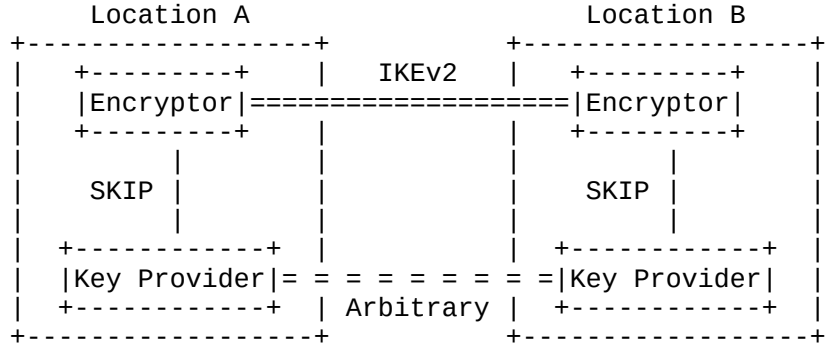


Figure 1: SKIP Network Overview

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. Protocol Overview

SKIP defines the interface through which two encryptors can obtain a key from their co-located Key Providers. The diagram Figure 2 provides an overview of the steps involved.

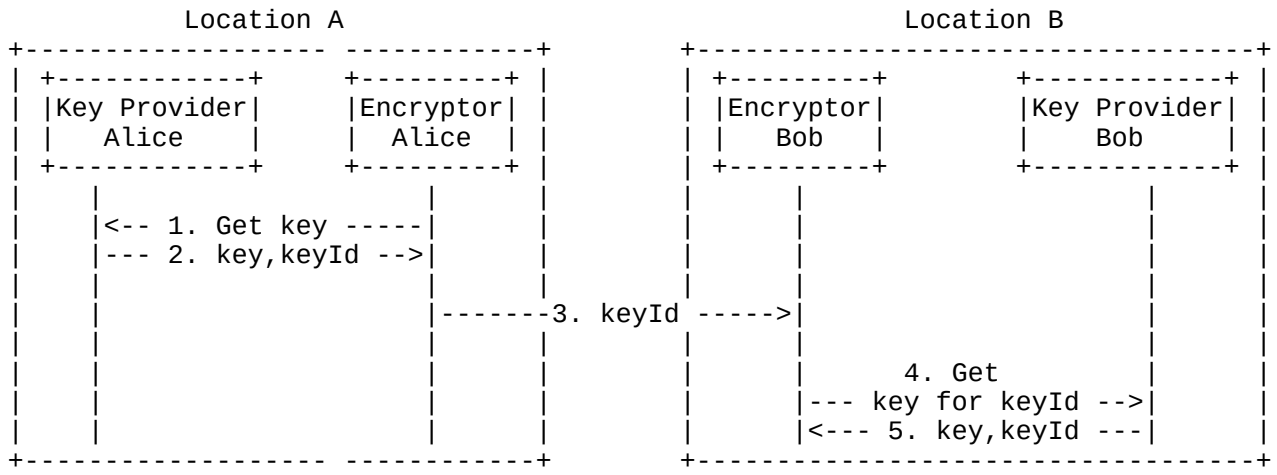


Figure 2: SKIP Key Exchange

1. Encryptor Alice initiates a request to KP Alice for a key.

2. KP Alice generates a key and a unique identifier keyId, synchronizes the key and keyId with KP Bob through an arbitrary protocol, returns the key and keyId to encryptor Alice, and finally zeroizes the local copy of the key.

3. Encryptor Alice establishes a connection with encryptor Bob and exchanges the keyId.

4. Encryptor Bob initiates a request to KP Bob for the key associated with the keyId provided by encryptor Alice.

5. KP Bob responds with the key associated with the keyId and zeroizes its local copy.

At the end of this exchange, encryptors Alice and encryptor Bob possess the same key, which can be utilized as a pre-shared key in another protocol, thereby enhancing its security against potential quantum threats.

3. SKIP Interface

The connection between the Key Provider and the encryptor is established using IP over Ethernet. The communication protocol used is Hypertext Transfer Protocol (HTTP) over Transport Layer Security (TLS) as per [RFC9110], with TLS versions 1.2 or 1.3 [RFC8446]. The table below lists supported TLS ciphersuites and authentication modes.

Mode	TLS version	Ciphers (Algorithm)	Requirement
Certificate	TLS 1.2	TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	RECOMMENDED
Pre-Shared Key (PSK)	TLS 1.2	TLS_DHE_PSK_WITH_AES_256_CBC_SHA384, TLS_DHE_PSK_WITH_AES_256_CBC_SHA	RECOMMENDED
Certificate/PSK	TLS 1.3	TLS_AES_256_GCM_SHA384	REQUIRED

Table 1: TLS Authentication Modes.

4. SKIP Methods and Status Codes

An encryptor uses the following methods to interact with a Key Provider:

NO.	Method	Path	Meaning
1.	GET	https://{host-identifier:port}/capabilities	Get the capabilities of the KP
2.	GET	https://{host-identifier:port}/key?remoteSystemID=Bob	Get a key that is shared with KP having localSystemID Bob
3.	GET	https://{host-identifier:port}/key?remoteSystemID=Bob&size=128	Get a key of size 128 bits that is shared with KP having localSystemID Bob
4.	GET	https://{host-identifier:port}/key/{keyId}?remoteSystemID=Alice	Get the key for the specified keyId that is shared with KP having localSystemID Alice
5.	GET	https://{host-identifier:port}/entropy	Get a random string having the default length of 256 bits
6.	GET	https://{host-identifier:port}/entropy?minentropy=128	Get a random string having the specified length of 128 bits

Table 2: SKIP Methods.

The host-identifier is an IPv4/v6 address or a hostname providing HTTPS services on standard port 443 or any port in the user-defined range. A KP SHOULD return one of the following HTTP status codes in response to a request made by an encryptor:

Code	Meaning
200	No problems were encountered
404	A path that doesn't correspond to those described in Table 2 was provided
405	A bad method was used. Only 'GET' is supported

Table 3: General Status Codes.

Data in the HTTP response from a KP to an encryptor is encoded in JSON format as described in [RFC8259].

4.1. Get Capabilities

Get capabilities method returns a JSON response detailing the capabilities of the KP. It provides an encryptor with an overview of services supported by the KP.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "entropy": {
      "type": "boolean"
    },
    "key": {
      "type": "boolean"
    },
    "algorithm": {
      "type": "string"
    },
    "localSystemID": {
      "type": "string"
    },
    "remoteSystemID": {
      "type": "array",
      "items": [
        {
          "type": "string"
        },
        {
          "type": "string"
        }
      ]
    }
  },
  "required": [
    "entropy",
    "key",
    "algorithm",
    "localSystemID",
    "remoteSystemID"
  ]
}
```

Figure 3: The schema of get capabilities response body.

```
{
  "entropy" : true,
  "key" : true,
  "algorithm": "PRF",
  "localSystemID": "Alice",
  "remoteSystemID": [
    "Bob",
    "Eve"
  ]
}
```

Figure 4: An example get capabilities response body.

The fields returned in the capabilities JSON response are as follows:

Field	Description
entropy	True if the KP supports the GET /entropy method
key	True if the KP supports the GET /key method
algorithm	Identifier or description of the algorithm used by the KP for generating and synchronizing keys
localSystemID	Identifier or name associated with the KP
remoteSystemID	List of identifiers of remote KPs with which the queried KP can synchronize a key

Table 4: SKIP capability response fields.

The frequency at which an encryptor requests the capabilities of its co-located KP can depend on the expected frequency of changes in the KP network. If the KP supports dynamically learning of a newly onboarded KP, then an encryptor MAY download the capabilities on each SKIP execution to ensure it receives an up-to-date remoteSystemID list. Conversely, if the KP network is unlikely to change, an encryptor MAY download the capabilities only once and cache the results. Implementation MUST support at least 16 bytes for Algorithm field and 32 bytes for localSystemID and remoteSystemID.

4.1.1. Local SystemID

Each KP is associated with an identifier, represented by the localSystemID field in the capabilities response. The uniqueness of this identifier is crucial if there are multiple KPs connected to a single encryptor, or if a single KP is communicating with multiple peer KPs. An implementor can choose the scope of the uniqueness of this identifier to be either global or connection-specific.

1. Global Uniqueness: The identifier is unique to all the KPs within the network.

2. Connection-Specific Uniqueness: The identifier is unique among all the KP connections attached to the encryptor. This option is available only if the KP can only share keys with exactly one other KP (and vice versa).

4.1.2. Remote SystemID

The remoteSystemID field contains a list of identifiers for KPs with which the queried KP can synchronize a key. At least one identifier MUST be specified in the list. A list entry MAY use a glob pattern to represent more than one KP identifier. This feature can shorten the length of the remoteSystemID list in large networks with numerous KPs. For example, two KP identifiers KP_ALICE_LOC1 and KP_BOB_LOC1 can be expressed with a single list entry KP_*_LOC1. The following glob patterns are supported in accordance with [POSIX] standards:

Pattern	Description
*	matches multiple characters
?	matches any single character
[list]	matches any single character in the list
[!list]	matches any single character not in the list

Table 5: SKIP remote system id glob patterns.

4.2. Get Key

Get key method returns a JSON response containing a key along with a corresponding key identifier. It facilitates the delivery of a key from a KP to an encryptor.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "keyId": {
      "type": "string"
    },
    "key": {
      "type": "string"
    }
  },
  "required": [
    "keyId",
    "key"
  ]
}
```

Figure 5: The schema of the get key response body.

```
{
   "keyId" : "1726e9AE76234FB1dd1283d4dca1911e1f93864d70f3069e",
   "key" : "ad229dfb8a276e74c1f3b6c09349a69fb2fed73c541270663f0e5cbbfb031670"
}
```

Figure 6: An example get key response body.

The fields returned in the key response are as follows

Field	Description
keyId	Hexadecimal-encoded identifier string associated with the key
key	Hexadecimal-encoded bytes of the key string

Table 6: SKIP key response fields.

An encryptor can request a (keyId, key) pair, or a key associated with a specific keyId. In the typical SKIP flow, the initiating encryptor will request a fresh (keyId, key) pair from its co-located KP and then pass the keyId to the responder encryptor. The responder encryptor will then retrieve the key from its co-located KP using the given keyId. To request a specific key, the hexadecimal-encoded keyId is specified within the URL along with the remoteSystemID as outlined in method 4 of Table 2. An encryptor can also request keys of a specific bit size by encoding the size within the URL as outlined in method 3 of Table 2.

A KP SHOULD return the following HTTP status codes in response to key request by an encryptor.

Code	Meaning
200	OK
400	A malformed keyId was requested or the key was not found
500	There was an internal error while trying to read or zeroize the key

Table 7: Get key Status Codes.

4.2.1. keyId

Each key supplied by a KP is associated with a unique key identifier. The bit position in a keyId uniquely maps only to a particular key, guaranteeing that any key request for a specific keyId always yields the same key. The key MUST NOT be recoverable with knowledge of the keyId alone. An encryptor in possession of a valid keyId can use it to request its associated KP for the corresponding key. The keyId is returned in responses and supplied in requests as a hexadecimal-encoded string and has a default length of 128 bits.

4.2.2. Key

The key bytes, returned as a hexadecimal-encoded string have a default length of 256 bits. A KP MUST zeroize its local copy of a key after it is provided to an encryptor.

4.3. Get Entropy

Get entropy method returns a JSON response containing a randomly generated entropy string and the length of this string in bits. encryptors can request an entropy sample for its internal consumption. The KPs MUST NOT utilize the entropy sample for any other purpose.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "randomStr": {
      "type": "string"
    },
    "minentropy": {
      "type": "integer"
    }
  },
  "required": [
    "randomStr",
    "minentropy"
  ]
}
```

Figure 7: The schema of the get entropy response body.

```
{
   "randomStr" : "AD229DFB8A276E74C1F3B6C09349A69FB2FED73C541270663F0E5CBBFB031670",
   "minentropy" : 256
}
```

Figure 8: An example get entropy response body.

The default length of the entropy supplied by randomStr field is 256 bits. An encryptor can request an entropy sample of a specific bit size by encoding the minentropy size within the URL as outlined in method 6 of Table 2.

Field	Description
randomStr	Hexadecimal-encoded random bytes string
minentropy	Length of random bytes provided in response

Table 8: SKIP entropy response fields.

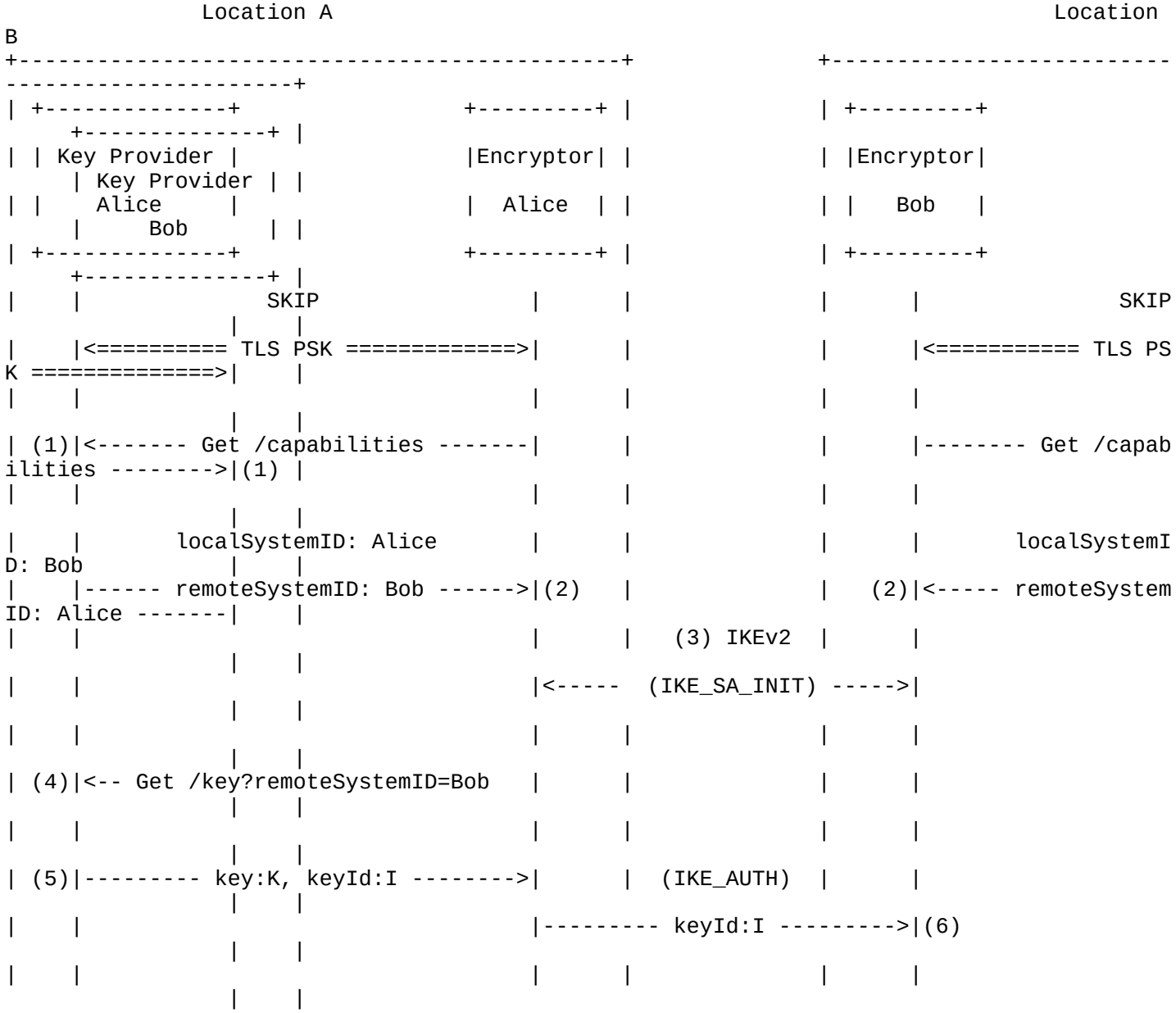
A KP SHOULD return the following HTTP status codes in response to entropy request by the encryptors.

Code	Meaning
200	OK
503	A hardware random number generator is not available or the entropy pool doesn't have enough entropy bits

Table 9: Get entropy Status Codes.

5. SKIP with IKEv2

SKIP can be used to dynamically supply post-quantum pre-shared keys (PPKs) in the IKEv2 PPK protocol extension [RFC8784]. The process of how SKIP is utilized in this context is outlined below.



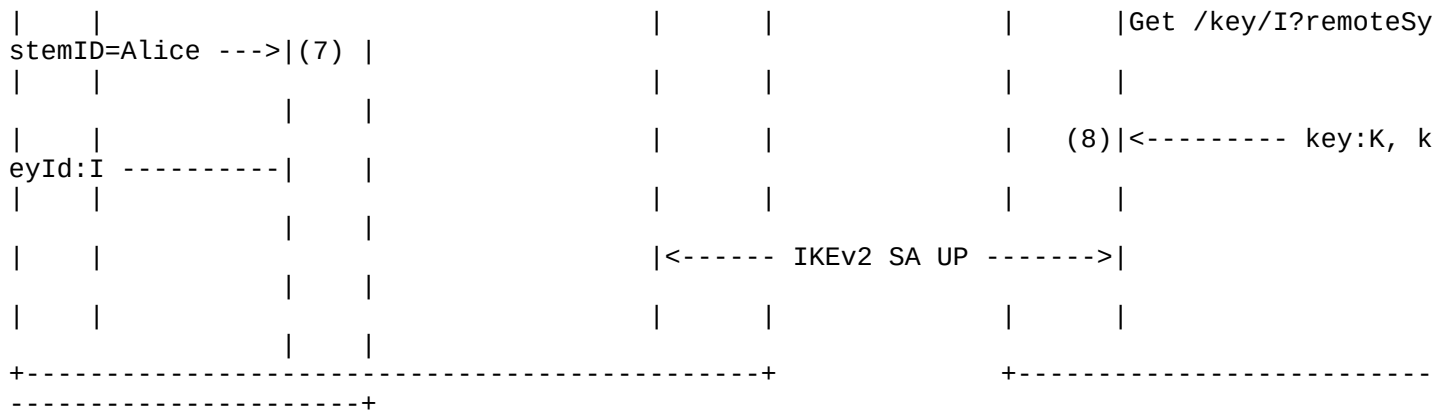

| | | | | |Get /key/I?remoteSy
stemID=Alice --->|(7) |
| | | | | |
| | | | | (8)|<--------- key:K, k
eyId:I ----------| |
		<------ IKEv2 SA UP ------->
+---+   +-------------------------
----------------------+

Figure 9: SKIP Protocol Exchange

(1) Each encryptor establishes a secure TLS connection with its corresponding Key Provider and retrieves the capabilities of the co-located Key Provider using the get capabilities method (refer Section 4.1).

(2) The Key Provider responds with its capabilities. For simplicity, only the localSystemID and remoteSystemID fields are shown in the capabilities response in the diagram.

(3) As part of IKE_SA_INIT exchange, encryptors propose the use of IKEv2 PPK extension by including USE_PPK notification payload, having type 16435, protocol ID of 0, no Security Parameter Index (SPI), and the localSystemID of its co-located Key Provider as notification data.

(4) Encryptor Alice invokes the get key method with Key Provider Alice to obtain a (keyId, key) pair (refer Section 4.2). The key request URL is encoded as https://{host-identifier:port}/key?remoteSystemID=Bob.

(5) Key Provider Alice responds with a key and its associated keyId.

(6) Encryptor Alice transmits the keyId to the peer encryptor Bob as part of IKE_AUTH request message using PPK_IDENTITY notification payload, having type 16436, protocol ID of 0, no SPI, and PPK_ID Type as 2 (PPK_ID_FIXED) followed by keyId as notification data.

(7) Encryptor Bob invokes the get key method with Key Provider Bob to retrieve the key associated with the keyId. The key request URL is encoded as https://{host-identifier:port}/key/{keyId}?remoteSystemID=Alice.

(8) Key Provider Bob responds with the key associated with the keyId.

At the end of this exchange, both encryptors possess the identical key, which is utilized to create key material for the IKEv2/IPsec Security Associations (SAs). [QRIPSEC]

Although this document uses IKEv2 with SKIP as an example, it’s important to note that SKIP is a generic protocol that can be integrated with any existing security protocols by adding suitable extensions to provide quantum resistance.

6. SKIP Use Cases

This section of the document describes the use cases where SKIP can be leveraged and deployed, particularly in scenarios requiring robust network encryption across a wide range of industries.

6.1. High Speed Data Center Interconnect

High speed data center interconnection (DCI) connects multiple data centers using high speed connectivity. The DCI encryption use case caters to industries that require secure, high speed data transport between multiple data centers for critical operations such as disaster recovery backups, synchronous and asynchronous replication, and extending data center fabric in a private data center or within a colocation space.

6.1.1. Industries covered

Industries such as telecommunications carriers, financial institutions, cloud service providers, large enterprises, government entities, and defense agencies. For these sectors, maintaining high uptime is critical, along with the protection and security of data transmitted across this infrastructure. The need for robust encryption is driven by the sensitive nature of the data and the potential consequences of data breaches or interruptions.

6.1.2. Common topology(s)

The topologies typically encountered in this use case are point-to-point (p2p), where a direct p2p link is established between two data centers. This topology is favored for its simplicity and efficiency in handling large volumes of data traffic.

6.1.3. Encryption types

For DCI use case, p2p topologies align well with IEEE 802.1AE MAC Security (MACsec) due to its simplicity and the capability to support high-speed transport encryption at link rates exceeding 400 Gbps. SKIP can be integrated with MACsec to provide dynamic key management and enhance the security of the key exchange process. In some cases needing the flexibility of IP transport, IPsec is applicable, although in a smaller subset of use cases. IPsec operates at the network layer and can secure data across diverse network paths. SKIP can be integrated with IPsec to provide PPKs which provide resistance to quantum computing attacks.

6.2. Access and Aggregation Backhaul networks

Backhaul networks refer to the aggregation of branch and remote sites to a centralized location. These use cases and topologies are very common for both enterprise and service provider networks that require access to resources and applications typically hosted at a centralized location (i.e., private data centers, colocation facilities, and more recently inside of public clouds). They are indispensable for industries that require reliable and secure connectivity from multiple locations typically over lower-cost public IP transport (i.e., Internet, 5G, LEO). For high speed requirements, private Metro Ethernet transport services can be leveraged.

6.2.1. Industries covered

Backhaul networks are essential for industries like telecommunication, service providers, enterprises, government, and commercial entities for secure access to vital resources relevant to the mission and business.

6.2.2. Common topology(s)

Topologies found in this use case typically include point-to-point connections in a hub-and-spoke topology, but can also support other diverse topologies such as point to multipoint, full/partial-mesh, or ring configurations, depending on the network topology, redundancy and security requirements.

6.2.3. Encryption types

IPsec is utilized for securing site-to-site connections in the various topologies, enabling secure data transmission between branch offices to a central corporate networks and data centers, or to and from a data center or colocation facility.

MACsec is typically implemented when Metro Ethernet backhaul is leveraged to provide the high-speed encryption capabilities needed to secure point-to-point or point-to-multipoint connections over these high-speed transport options.

6.3. Cloud service providers

Cloud service providers (CSPs) deliver infrastructure, applications and workload management services to a wide spectrum of industries. These industries entrust their sensitive and confidential data to CSPs, which necessitates secure handling of data during transit or at rest. While the ability exists today for securing the private link connection from a CSP partner into the CSP environment, the ability

for operators to leverage the global infrastructure of the CSP for optimal and cost-effective inter and intra-region transport is becoming more common as a WAN transport alternative.

6.3.1. Industries covered

Enterprise, financial services, healthcare, education and government entities are some of the top industries that utilize the cloud service providers.

6.3.2. Common topology(s)

Topologies in this use case are varied and include point-to-point, hub and spoke, full/partial mesh, or a hybrid approach combining elements of the aforementioned topology types.

6.3.3. Encryption types

IPsec is the most common type of encryption used to securely route traffic from customer network to the cloud service provider network, or for the intra/inter-region design options aforementioned above. MACsec is another option for those operators wanting to leverage a high-speed private link from the enterprise into the private domain of the CSP, and this form of secure high-speed connectivity into the cloud is available today from some public CSP’s.

6.4. Scale target

SKIP is designed to be both flexible and scalable, making it suitable for networks ranging from small-scale to large-scale. It enables the provision of post-quantum security without requiring an overhaul of the existing encryption framework.

6.5. SKIP usage

SKIP can be utilized across all the use cases and delivers all the benefits highlighted in this document. It allow operators to leverage external QKD or PQC cloud-based key sources and benefit from the automated provisioning, refresh, and entropy of the imported PPK, either for MACsec or IPsec.

7. IANA Considerations

This document updates the use of the USE_PPK (16435) notify message as defined in [RFC8784] to include the localSystemID of the Key Provider as notification data.

8. Security Considerations

SKIP is designed to facilitate the secure distribution of keys over a network. Its security depends primarily on two considerations: the strength of keys generated by the Key Provider (KP) and the secure delivery of keys from KPs to encryptors.

For the first consideration, this document does not impose any restrictions on the mechanism used by a KP to generate the key. In general, the same security considerations for generating a post-quantum pre-shared key (PPK) outlined in [RFC8784] apply equally here. In particular, it is strongest practice to ensure that a key generated by a KP has at least 256 bits of entropy, which will provide 128 bits of post-quantum security when Grover's algorithm [GROVER] is taken into account.

For the secure delivery of keys within SKIP, there are three different links (physical or logical) to consider: (1) the link between the two KPs, (2) the link between the two encryptors, and (3) the link between a KP and an encryptor. We will address each in turn. For (1), the mechanism by which two KPs synchronize a key is intentionally out-of-scope for SKIP, such that it can interoperate with various hardware or software technologies. It should be clear, however, that this key synchronization mechanism should be quantum-resistant if the key is intended to upgrade an existing protocol to quantum resistance. To this end, KPs can employ one or more technologies believed to be quantum-resistant, including, but not limited to: post-quantum cryptography (PQC), quantum key distribution (QKD), a trusted third-party protocol, or a one-time pad (OTP). For (2), this document makes no assumptions about the security of this link. Indeed, the primary purpose of SKIP is to augment an existing protocol between the two encryptors in the face of future quantum computing or other cryptanalytic advances. As such, the only SKIP-related piece of data to traverse this link is an opaque key identifier, from which it MUST be infeasible to derive the corresponding key. After the SKIP exchange completes, the two encryptors can use the key as a pre-shared key. The link in (3) between the KP and encryptor is specified in Table 1 to be HTTP over TLS v1.2 or v1.3, with support for both certificate and pre-shared key (PSK) authentication modes of TLS. The inclusion of certificates based on Rivest-Shamir-Adelman (RSA) and elliptic curve cryptography (ECC) that are known to have vulnerabilities to quantum computers recognizes the reality of interoperating with encryptors that do not yet have access to post-quantum cryptography, in addition to the the lack of post-quantum x509 certificates at the time of writing. The use of PSK-based authentication is RECOMMENDED since it is believed to be quantum-resistant, though the use of PSKs can introduce the same administrative challenges that SKIP is trying to solve for the

encryptor-to-encryptor link. To mitigate these issues, an implementor SHOULD seek to limit the exposure of the KP-to-encryptor link by co-locating the KP and encryptor, and employ techniques such as network segmentation. Where feasible, running the KP on the same physical device as the encryptor as a co-process or hosted application can also significantly reduce the exposure of this link. Post-quantum key exchange algorithms MAY also be used to secure this link when they are widely deployed.

Finally, there is an additional security consideration that the identifier scheme used for KPs can potentially leak information about the larger network topology or about specific software or hardware versions in use. In particular, access to the remoteSystemID list in the KP capabilities response may help an adversary in finding lateral compromises within a network or new insertion points into the network. To mitigate this threat, an identifier scheme based on random pseudonyms can remove any correspondence between the KP and its location or underlying technology. The use of the glob patterns in the remoteSystemID field can also conceal specific details about the KP network by collapsing groups of KPs into a single entry in the remoteSystemID list.

9. References

9.1. Normative References

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/rfc/rfc8784>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

9.2. Informative References

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[GROVER] Grover, L. K., "A Fast Quantum Mechanical Algorithm for Database Search, STOC ’96: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, pp. 212-219", July 1996, <https://doi.org/10.1145/237814.237866>.

[QRIPSEC] Hill, C., Kawaguchi, S., and J. Lupo, "Engineering Quantum Resistance: An IPsec Case Study", February 2024, <https://www.qusecure.com/resources/ipsec-case-study-with-cisco-core-networking/>.

[POSIX] "IEEE/ISO/IEC International Standard - Information technology Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7, in ISO/IEC/IEEE 9945:2009(E)", September 2009, <https://doi.org/10.1109/IEEESTD.2009.5393893>.

[PQCRYPTO] "National Security Agency | Frequently Asked Question - Quantum Computing and Post-Quantum Cryptography", August 2021, <https://media.defense.gov/2021/Aug/04/2002821837/-1/-1/1/Quantum_FAQs_20210804.PDF>.

[CSFC] "National Security Agency Central Security Service - Commercial Solutions for Classified (CSfc) Symmetric Key Management Requirements Annex V2.1", May 2022, <https://www.nsa.gov/Portals/75/documents/resources/everyone/csfc/capability-packages/Symmetric%20Key%20Management%20Requirements%20v2_1.pdf>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/rfc/rfc4107>.

Acknowledgements

David McGrew, Scott Fluher, Lionel Florit, and Amjad Inamdar made significant contributions to this work.

Contributors

David McGrew
Cisco Systems, Inc.
Email: mcgrew@cisco.com

Scott Fluhrer
Cisco Systems, Inc.
Email: sfluhrer@cisco.com

Amjad Inamdar
Cisco Systems, Inc.
Email: amjads@cisco.com

Authors' Addresses

Rajiv Singh (editor)
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
United States of America
Email: rajisin2@cisco.com

Craig Hill
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
United States of America
Email: crhill@cisco.com

Scott Kawaguchi
QuSecure, Inc.
1900 South Norfolk Street, Suite 350/11
San Mateo, CA 94403
United States of America
Email: scott@qusecure.com

Joey Lupo
QuSecure, Inc.
1900 South Norfolk Street, Suite 350/11
San Mateo, CA 94403
United States of America

Email: jlupo@qusecure.com