Quantum-safe IPsec

Aliro

www.alirotech.com

Quantum-safe IPsec

Summary

This white paper provides an in-depth explanation of how to make IPsec quantum-safe, as well as step-by-step examples of how to implement PQC and QKD using popular vendor products for IPsec.

Introduction

Many of the cryptographic algorithms used to secure network communications today are vulnerable to quantum attacks. Internet Protocol Security (IPsec) depends on algorithms such as Diffie-Hellman, RSA, and Elliptic Curve Cryptography. All of these algorithms can be broken by a cryptographically relevant quantum computer. In the United States, NIST has published a draft report that disallows these quantum-vulnerable asymmetric algorithms in 2035. Other governments in other countries have put similar requirements in place.

Organizations in finance, healthcare, government, and defense should act now to protect IPsec from this emerging threat. Post-Quantum Cryptography (PQC) and Quantum Key Distribution (QKD) provide quantum-resilient protection, while hybrid approaches that combine PQC, QKD, and classical cryptography offer layered protection while meeting current compliance requirements. Upgrading to quantum-safe IPsec will ensure encrypted communications remain secure as quantum computing advances.

Practical implementation strategies, supported by standardized extensions and commercially available security products, help organizations strengthen their IPsec defenses. Proactively adopting these solutions will protect critical data from future quantum attacks, as well as risks due Harvest Now Decrypt Later (HNDL) attacks.

What is Internet Protocol Security (IPsec)?

Internet Protocol Security, or IPsec, is a collection of security protocols that enable the secure exchange of messages over some untrusted network, for example, the public internet. it operates at the IP layer in the network stack. These security measures include confidentiality, authentication, integrity, replay protection, compression, Network Address Translation (NAT) Traversal, and Traffic Flow Confidentiality (TFC). Within this framework, the services that are directly related to quantum safety are:

- **Confidentiality.** Encrypting data so only authorized parties can decrypt it.
- **Authentication.** Verifying the identity of the sender to the receiver.
- **Integrity.** Ensuring data has not been altered in transit between the sender and receiver.

The most prevalent application for IPsec is Virtual Private Networks. There are two types of VPNs: Gateway-to-Gateway VPNs and Host-to-Gateway VPNs. Gateway-to-Gateway VPNs connect multiple corporate networks. For example, headquarters and branch offices over some insecure IP network using encrypted IP terminals. Host-to-Gateway VPNs connect remote workers to some corporate network over the public Internet using encrypted IPsec tunnels.

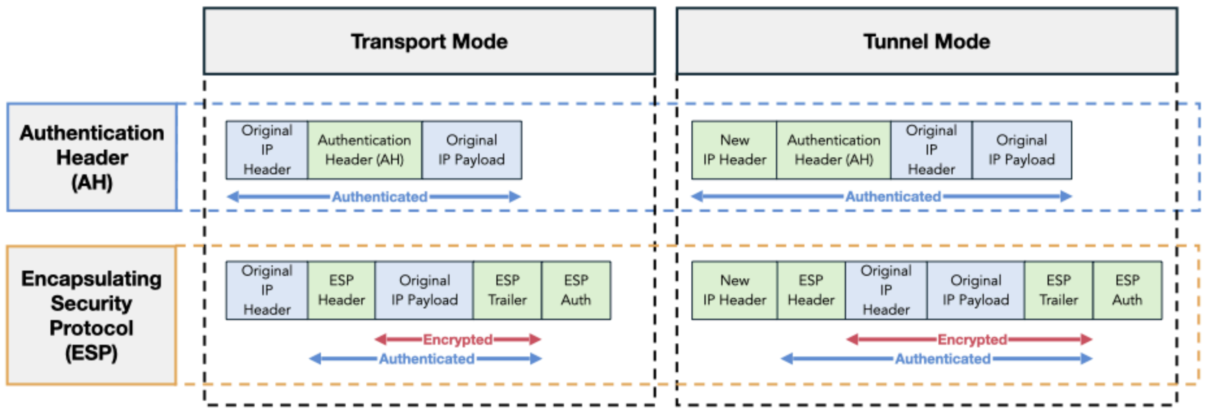
IPsec operates in two possible modes. Transport mode encrypts only the payload of the IP packet, leaving the original IP header intact. It is mainly used for secure host-to-gateway communication. Tunnel mode encrypts the entire IP packet (header and payload), encapsulating it within a new IP packet, and adding an additional IP header on top of it. This mode is typically used in Virtual Private Networks (VPNs) to securely connect two networks.

This white paper will only cover one of these control plane protocols, namely the Internet Key Exchange Protocol Version 2, or IKEv2. This is the most recent version and it's also the version necessary for deploying quantum-safe networks.

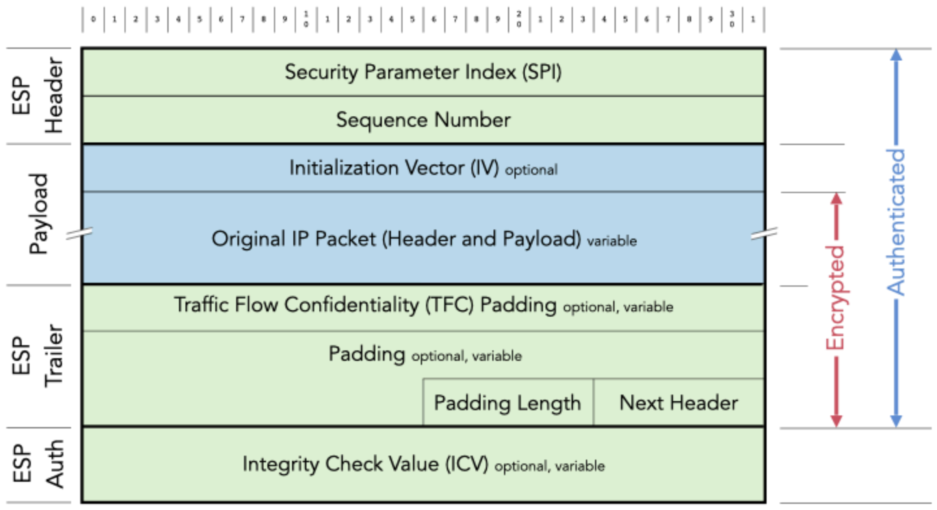
IPsec Protocols

IPsec is not one single protocol. Instead, it's a very large and complex suite of protocols. There are two data plane protocols that secure the user data IP packets, namely authentication headers and encapsulating security payload (ESP). There are also multiple control pane protocols that are responsible for things like authentication, creating and managing security associations and for negotiating cryptographic parameters. Authentication headers only do authentication and integrity protection, but not encryption. Encapsulating security payload also does encryption.

Each of the four possible combinations (transport mode + AH, transport mode + ESP, tunnel mode + AH, and tunnel mode +ESP) has its own specific use case, but here we will focus on ESP tunnel mode, which is the option in the bottom right pictured below.



The diagram below shows the format of a data plane ESP packet in tunnel mode:


ESP Header
Payload
ESP Trailer
ESP Auth
Security Parameter Index (SPI)
Sequence Number
Initialization Vector (IV) optional
Original IP Packet (Header and Payload) variable
Traffic Flow Confidentiality (TFC) Padding optional, variable
Padding optional, variable
Padding Length
Next Header
Integrity Check Value (ICV) optional, variable
Encrypted
Authenticated

The security parameter index identifies the security association.

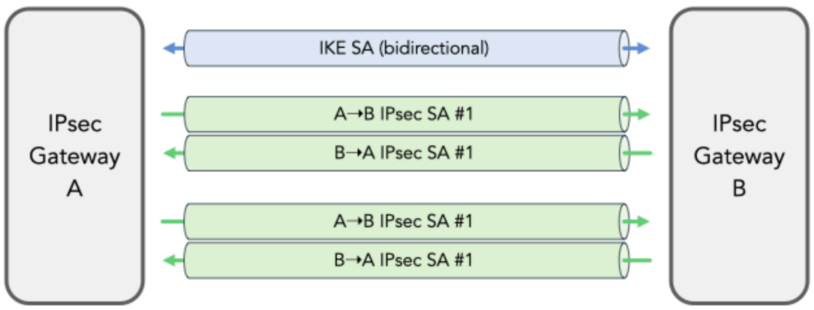
- The sequence number is used to prevent replay attacks.
- The original IP packets that are sent through the IPsec tunnel are encrypted and encapsulated into the ESP packets, as indicated by the blue arrow above.
- Some encryption algorithms also require an initialization vector, which is shown in the blue box above.
- The padding is used for alignment. It can also be used for something known as traffic flow protection, i.e. for hiding traffic patterns.
- Finally, the authentication trailer contains an integrity check value, which is used for authentication and integrity checking.

The concept of a Security Association is very important in IPsec. A Security Association is a set of cryptographic parameters agreed upon between the two communicating IPsec gateways. This set of negotiated cryptographic parameters includes the encryption protocol, the authentication mode, the Diffie-Hellman group, the pseudo-random function that's used for deriving keys, and the integrity algorithm that's used for generating digital signatures. Each of these concepts will be covered in more detail later on in this white paper.

Security Associations (SAs)

Set of agreed cryptographic parameters:

- The encryption algorithm (e.g. AES-128-GCM)
- The authentication method (pre-shared keys or certificates)
- The Diffie-Hellman group for key agreement (e.g. group 4).
- Pseudo-Random Function (PRF) (e.g. HMAC-SHA-1)
- The integrity algorithm, i.e. the digital signing algorithm (e.g. HMAC-SHA1-96).



There are typically multiple Security Associations within an IPsec tunnel. IKE Security Associations which, shown in blue above, are bidirectional and used for the IKEv2 control plane messages. IPsec Security Associations, shown in green above, are unidirectional. They usually appear in pairs and are used for ESP data plane messages. There are also often multiple IPsec Security Associations for different flows of traffic and also for re-keying.

The Internet Key Exchange protocol Version 2, or IKEv2, is the control plane protocol for IPsec. It is responsible for authenticating peers, for managing security associations, and for negotiating cryptographic parameters.

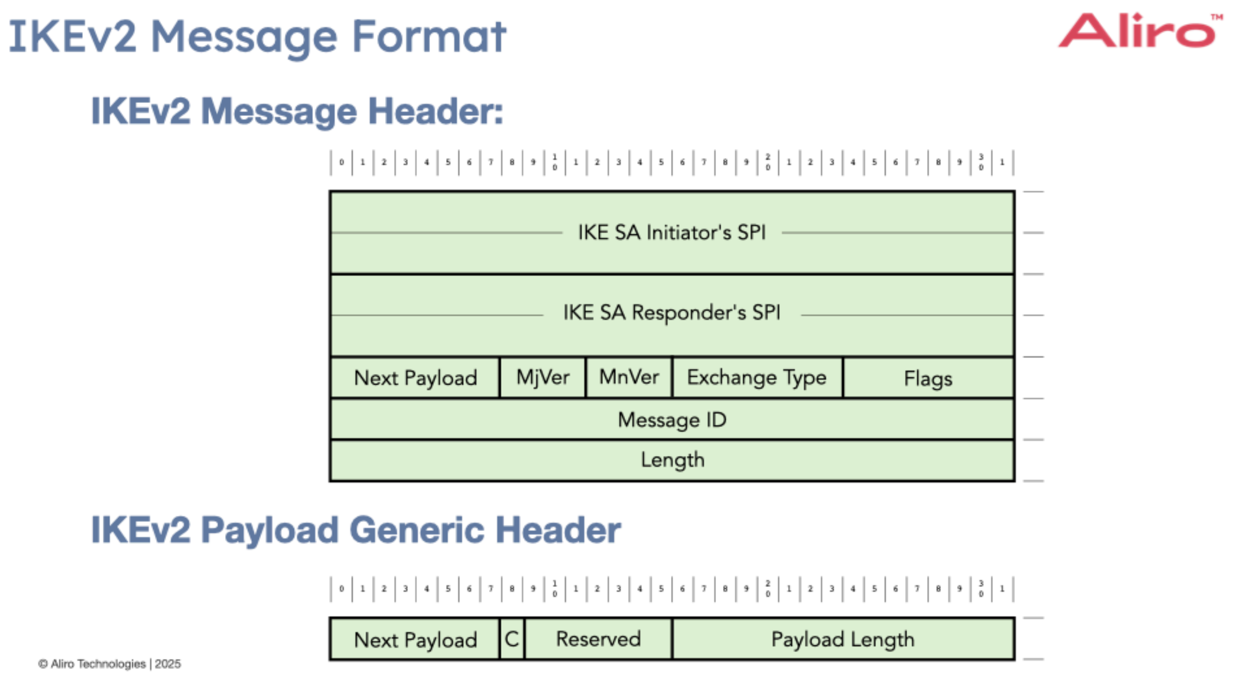
IKEv2 runs over UDP. Because UDP is unreliable, IKEv2 implements its own reliability mechanism using message IDs and retransmission timers.

Because IP fragmentation causes practical problems related to Network Address Translation (NAT), IKEv2 implements its own fragmentation mechanism to deal with messages that are larger than the path MTU.

IKEv2 messages always come in pairs, called exchanges, which consist of a request that is sent by the initiator and a response that is sent by the responder.

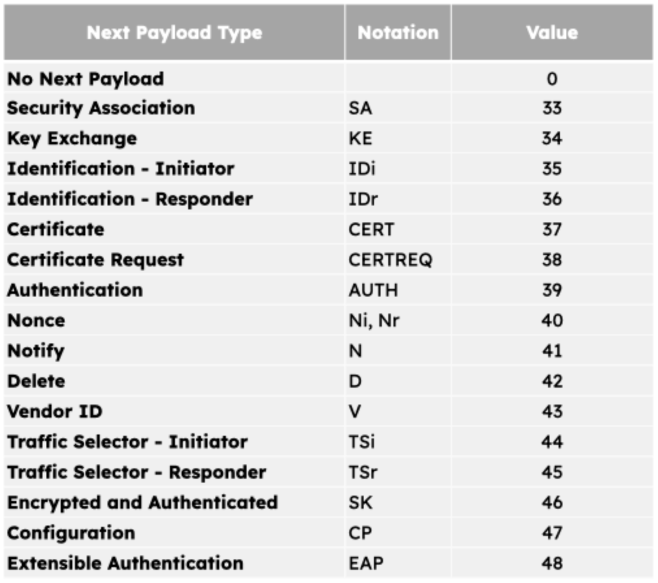
There are several exchange types:

- IKE_SA_INIT and IKE_AUTH exchanges are used to establish the initial IKE Security Association and the initial IPsec Security Association.
- CREATE_CHILD_SA exchanges are used to establish additional security associations, either for new traffic flows or for rekeying.
- INFORMATIONAL exchanges are used to tear down a security association, to report error conditions, and for other types of housekeeping.
- There are also some other exchange types that are defined in IPsec extensions, including quantum security extensions, discussed later in this white paper.



Pictured above is the format of an IKEv2 control plane packet. Note that there is always a fixed IKEv2 header which is shown at the top, which is then followed by a series of one or more IKEv2 payloads. Each payload is identified by a payload type and has its own generic header which is shown at the bottom.

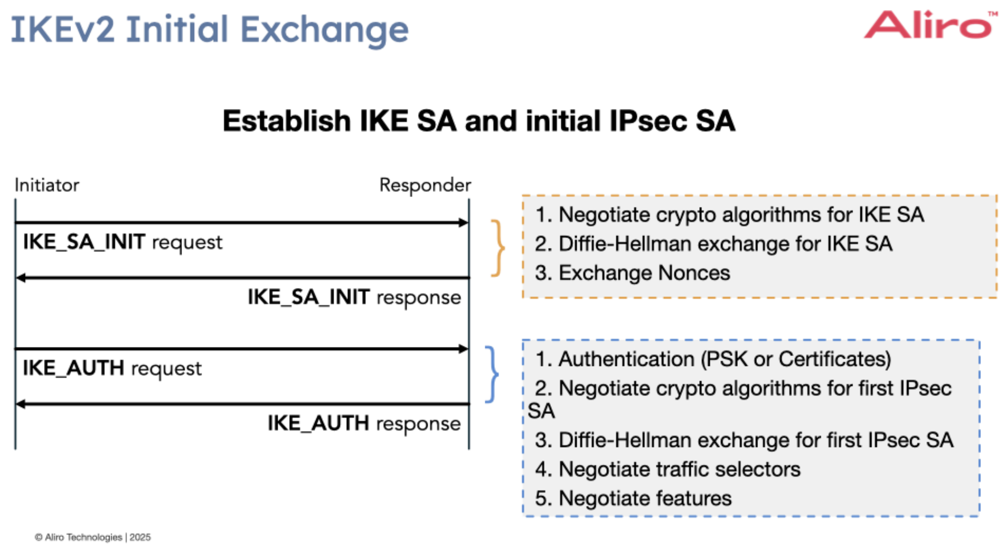
Below is a list of header types that are defined in the base IKEv2 standard.



Next Payload Type	Notation	Value
No Next Payload		0
Security Association	SA	33
Key Exchange	KE	34
Identification - Initiator	IDi	35
Identification - Responder	IDr	36
Certificate	CERT	37
Certificate Request	CERTREQ	38
Authentication	AUTH	39
Nonce	Ni, Nr	40
Notify	N	41
Delete	D	42
Vendor ID	V	43
Traffic Selector - Initiator	TSi	44
Traffic Selector - Responder	TSr	45
Encrypted and Authenticated	SK	46
Configuration	CP	47
Extensible Authentication	EAP	48

There are additional header types that are defined in IKEv2 extensions, including the new extensions for quantum secure IPsec.

The IKEv2 protocol always begins with an initial exchange. The initial exchange usually consists of an IKE_SA_INIT exchange shown at the top of the image pictured above, followed by IKE_AUTH exchange shown at the bottom, for a total of four messages. The purpose of an IKE_SA_INIT exchange is to negotiate the crypto algorithms, to do a Diffie-Hellman exchange to establish the session keys, and to exchange nonces.



IKEv2 Initial Exchange

Establish IKE SA and initial IPsec SA

1. Negotiate crypto algorithms for IKE SA
2. Diffie-Hellman exchange for IKE SA
3. Exchange Nonces

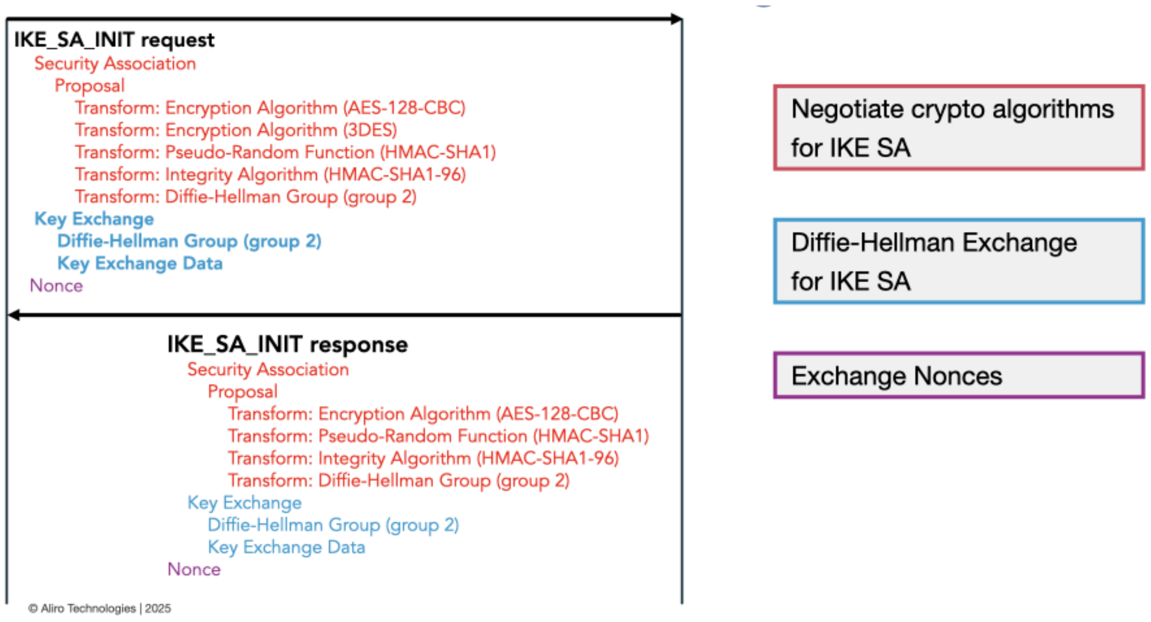
1. Authentication (PSK or Certificates)
2. Negotiate crypto algorithms for first IPsec SA
3. Diffie-Hellman exchange for first IPsec SA
4. Negotiate traffic selectors
5. Negotiate features

Then the IKE_AUTH exchange, which follows it, does several things.

1. It allows the two IP gateways to authenticate each other using either pre-shared keys or certificates.
2. It may optionally negotiate the cryptographic parameters for the first IPsec Security Association separately.
3. They may optionally do another Diffie-Hellman exchange for the first IPsec Security Association.
4. It will negotiate traffic selectors which identify the flows of traffic that are protected.
5. It negotiates the use of optional features. Each of these steps is outlined in more detail below.

The Initial IKE_SA_INIT Exchange

Pictured below is an initial IKE_SA_INIT exchange in more detail. The Security Association payload, highlighted in red, is used to negotiate the crypto algorithms for the control plane traffic in the initial IKE Security Association.



IKE_SA_INIT request
Security Association
Proposal
Transform: Encryption Algorithm (AES-128-CBC)
Transform: Encryption Algorithm (3DES)
Transform: Pseudo-Random Function (HMAC-SHA1)
Transform: Integrity Algorithm (HMAC-SHA1-96)
Transform: Diffie-Hellman Group (group 2)
Key Exchange
Diffie-Hellman Group (group 2)
Key Exchange Data
Nonce
IKE_SA_INIT response
Security Association
Proposal
Transform: Encryption Algorithm (AES-128-CBC)
Transform: Pseudo-Random Function (HMAC-SHA1)
Transform: Integrity Algorithm (HMAC-SHA1-96)
Transform: Diffie-Hellman Group (group 2)
Key Exchange
Diffie-Hellman Group (group 2)
Key Exchange Data
Nonce
Negotiate crypto algorithms for IKE SA
Diffie-Hellman Exchange for IKE SA
Exchange Nonces
© Aliro Technologies | 2025

The way it works is that the initiator sends an IKE_SA_INIT request, containing a list of proposals. The responder then selects the proposal that is preferred. Each proposal consists of a list of transforms, where each transform describes a particular crypto algorithm. So, for example, here we see that the initiator offers two options for the encryption algorithm, namely AES and triple DES. The responder chooses one of those options for encryption; pictured above the chosen option is AES. All other crypto parameters use the same negotiation mechanism.

The key exchange, which is highlighted in blue in the image above, is used to do a Diffie-Hellman exchange, which enables the two IPsec gateways to agree on a shared secret.

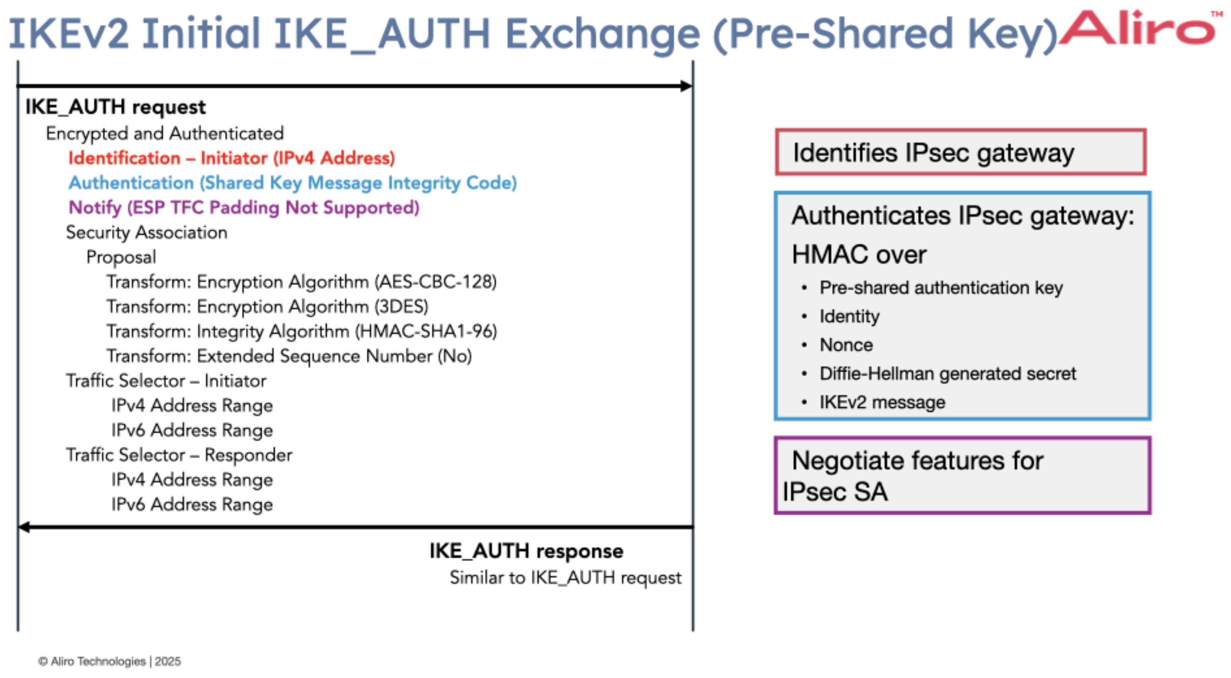
This shared secret is combined with nonces, identities, and other data as inputs into the key derivation function, which produces the various encryption keys and authentication keys that are used in IPsec. Diffie-Hellman will be covered in more detail later in this white paper.

Finally, the IKE_SA_INIT exchange is used to exchange nonces, highlighted in purple in the image above, and these nonces are random numbers that are mixed into the key derivation function to prevent replay attacks.

The Initial IKE_AUTH Exchange

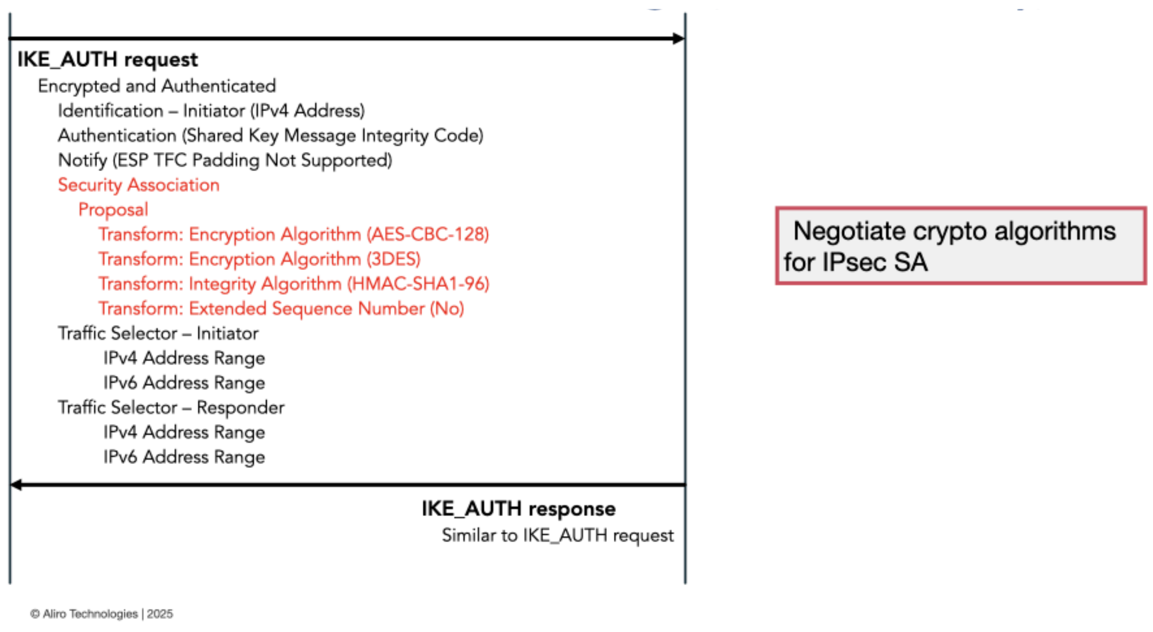
The second part of the initial exchange is the IKE_AUTH exchange.

The identification payload, highlighted in red below, is used to identify the IPsec gateways to each other. In this example, preshared keys are being used. IP addresses are used as identifiers. The authentication payload, highlighted below in blue, is used to authenticate the identity of the two IPsec gateways. In other words, it's used to prove that each IPsec gateway is indeed who they claim to be.


IKEv2 Initial IKE_AUTH Exchange (Pre-Shared Key)
Aliro™
IKE_AUTH request
Encrypted and Authenticated
Identification – Initiator (IPv4 Address)
Authentication (Shared Key Message Integrity Code)
Notify (ESP TFC Padding Not Supported)
Security Association
Proposal
Transform: Encryption Algorithm (AES-CBC-128)
Transform: Encryption Algorithm (3DES)
Transform: Integrity Algorithm (HMAC-SHA1-96)
Transform: Extended Sequence Number (No)
Traffic Selector – Initiator
IPv4 Address Range
IPv6 Address Range
Traffic Selector – Responder
IPv4 Address Range
IPv6 Address Range
IKE_AUTH response
Similar to IKE_AUTH request
Identifies IPsec gateway
Authenticates IPsec gateway: HMAC over
Pre-shared authentication key
Identity
Nonce
Diffie-Hellman generated secret
IKEv2 message
Negotiate features for IPsec SA
© Aliro Technologies | 2025

The authentication is performed by computing a hash based message authentication code, or HMAC for short, over several pieces of data, including the preshared authentication key and the identity. The responder then verifies the HMAC code. If the verification succeeds, it proves that both IPsec speakers have the same preshared authentication key, and it also proves that the message was not modified in transit. Later in this white paper, is an example of how authentication can also be accomplished using certificates.

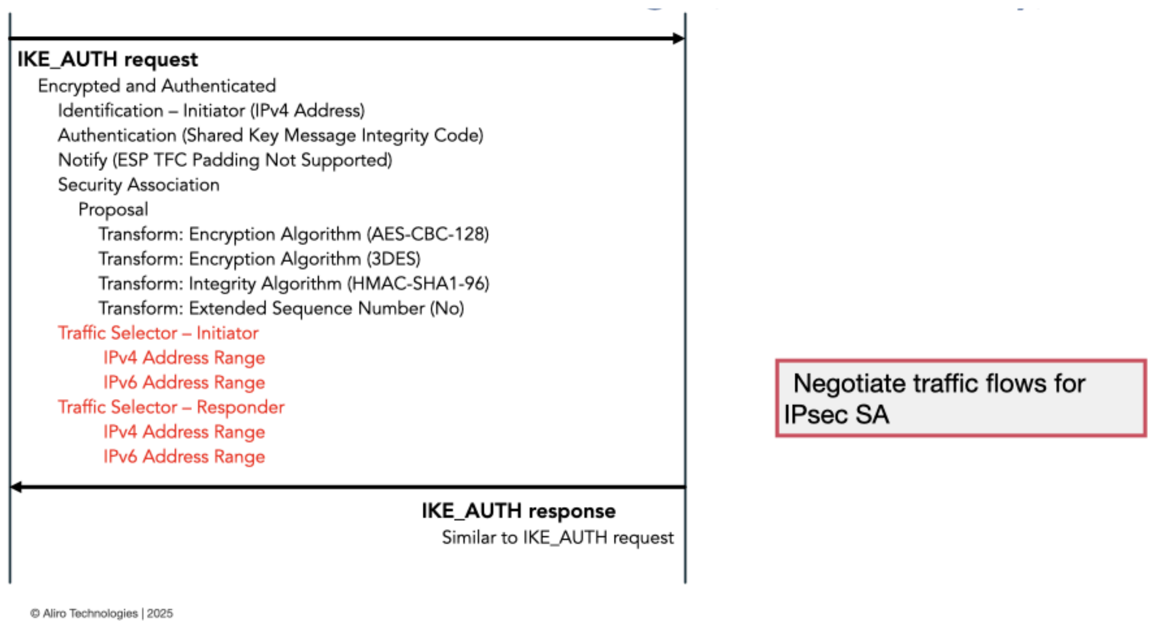
The IKE_AUTH message may also contain notify payloads, highlighted in purple in the image above. These are used to negotiate optional features. In this example, the use of traffic flow confidentiality padding is negotiated to not be used.



IKE_AUTH request
Encrypted and Authenticated
Identification – Initiator (IPv4 Address)
Authentication (Shared Key Message Integrity Code)
Notify (ESP TFC Padding Not Supported)
Security Association
Proposal
Transform: Encryption Algorithm (AES-CBC-128)
Transform: Encryption Algorithm (3DES)
Transform: Integrity Algorithm (HMAC-SHA1-96)
Transform: Extended Sequence Number (No)
Traffic Selector – Initiator
IPv4 Address Range
IPv6 Address Range
Traffic Selector – Responder
IPv4 Address Range
IPv6 Address Range
IKE_AUTH response
Similar to IKE_AUTH request
Negotiate crypto algorithms for IPsec SA
© Aliro Technologies | 2025

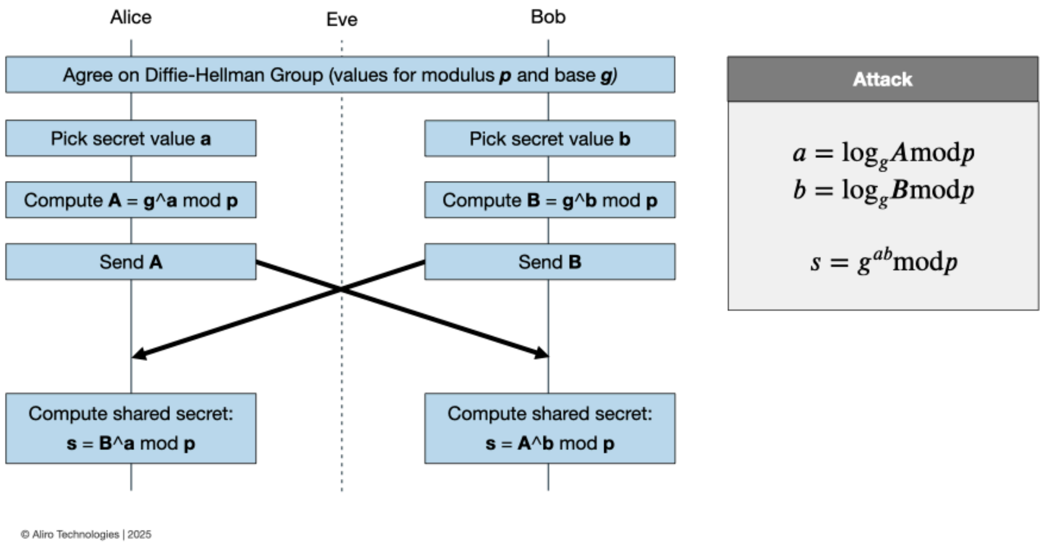
Highlighted in red above is another round of crypto algorithm negotiations similar to the negotiations before in the IKE_SA_INIT message, except in this case the negotiation is for the crypto algorithms for the data plane IPsec Security Association as opposed to the control plane IKE Security Association negotiated previously.

Finally the initiator and responder traffic selectors are used to negotiate which flows of traffic are encrypted, which flows of traffic flow through the IPsec tunnel, and these flows are identified by specifying a protocol, an address range and a port range, as seen in red in the image below.


IKE_AUTH request
Encrypted and Authenticated
Identification – Initiator (IPv4 Address)
Authentication (Shared Key Message Integrity Code)
Notify (ESP TFC Padding Not Supported)
Security Association
Proposal
Transform: Encryption Algorithm (AES-CBC-128)
Transform: Encryption Algorithm (3DES)
Transform: Integrity Algorithm (HMAC-SHA1-96)
Transform: Extended Sequence Number (No)
Traffic Selector – Initiator
IPv4 Address Range
IPv6 Address Range
Traffic Selector – Responder
IPv4 Address Range
IPv6 Address Range
IKE_AUTH response
Similar to IKE_AUTH request
Negotiate traffic flows for IPsec SA
© Aliro Technologies | 2025

The Diffie-Hellman Exchange

The purpose of the DiffieHellman exchange is for the two parties, which are labeled Alice and Bob in the image below, to agree on a shared secret using some public discussion. An eavesdropper, which has been called Eve here, should not be able to determine what that shared secret is, even if Eve is able to observe the entire public discussion.

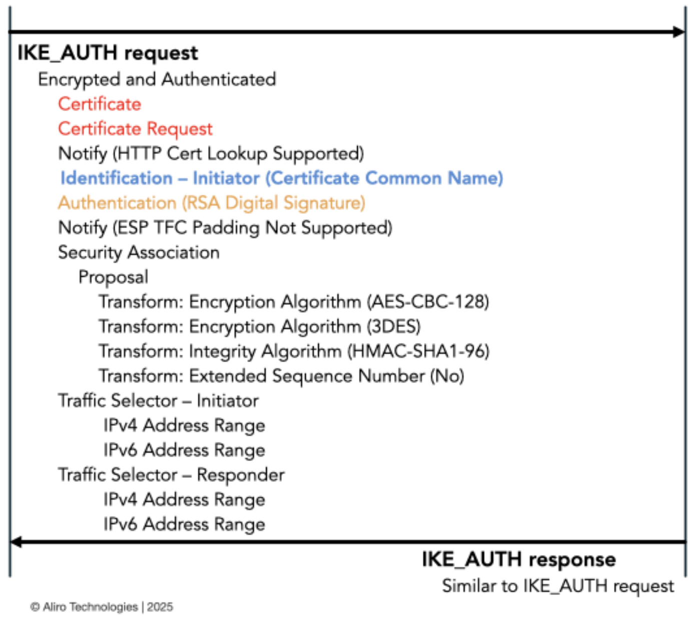


The two parties, Alice and Bob, perform a series of steps, shown in the diagram above, in order to agree on this shared secret. Some of these steps involve computing a modular exponentiation. For example, computing **g^a mod p**. These modular exponentiation operations are computationally easy to implement. Eavesdropper Eve is observing this public discussion and Eve can see certain values that are public (for example, the values p, g, A, and B) from those observed values. Eve could hypothetically reverse engineer to secret values, a and b, and thereby derive the shared secret, s. The way Eve would do that is shown on the gray box on the right in the image above.

The problem for Eve is that doing this reverse engineering involves computing what is known as a modular logarithm. This is the inverse of a modular exponentiation. Computing a modular logarithm is computationally infeasible on a classical computer, meaning that for sufficiently large values of p, it would literally take millions of years on even the biggest supercomputer available today to perform this kind of attack. The capability of future quantum computers to compute modular logarithms very efficiently using Shor's algorithm is discussed later in this paper. This capability is what makes Diffie-Hellman quantum vulnerable.

Certificate-based Authentication in IKE_AUTH

In addition to using preshared keys for authentication, IPsec also has another mechanism for doing authentication: certificates. Pictured below is an initial IKE_AUTH exchange that uses certificates instead of preshared keys for authentication.


IKE_AUTH request
Encrypted and Authenticated
Certificate
Certificate Request
Notify (HTTP Cert Lookup Supported)
Identification – Initiator (Certificate Common Name)
Authentication (RSA Digital Signature)
Notify (ESP TFC Padding Not Supported)
Security Association
Proposal
Transform: Encryption Algorithm (AES-CBC-128)
Transform: Encryption Algorithm (3DES)
Transform: Integrity Algorithm (HMAC-SHA1-96)
Transform: Extended Sequence Number (No)
Traffic Selector – Initiator
IPv4 Address Range
IPv6 Address Range
Traffic Selector – Responder
IPv4 Address Range
IPv6 Address Range
IKE_AUTH response
Similar to IKE_AUTH request
© Aliro Technologies | 2025

Certificate binds identity to public key

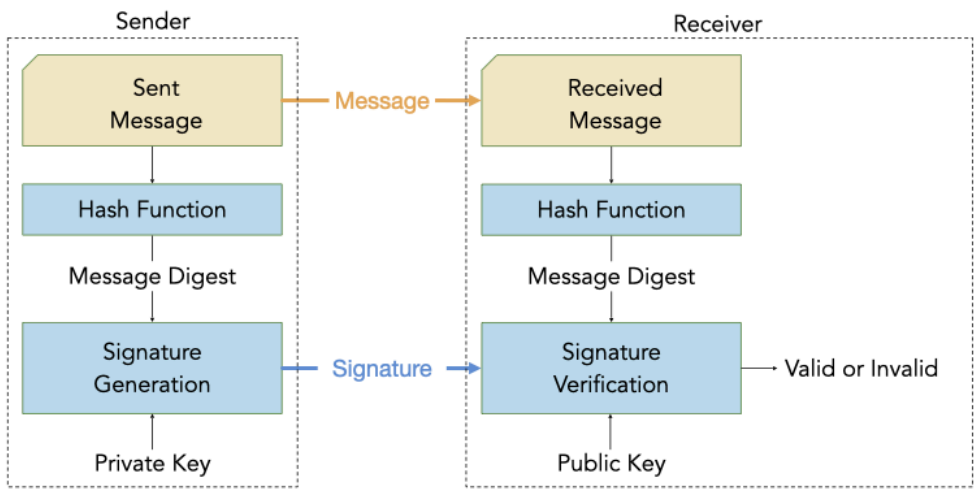
Identity is Common Name (CN) in certificate instead of IP address

Authentication uses Rivest Shamir Adleman (RSA) digital signature

There are three main differences between this method and the preshared key authentication method. First, the IKE_AUTH message includes a certificate and that certificate is used to bind a public key to an identity. Second, the identity is provided in the form of a common name instead of an IP address. Third, the authentication method is negotiated to use some type of digital signature algorithm, in this case negotiating the use of RSA instead of pre-shared keys.

Digital Signatures

In general, digital signatures follow the process pictured below.



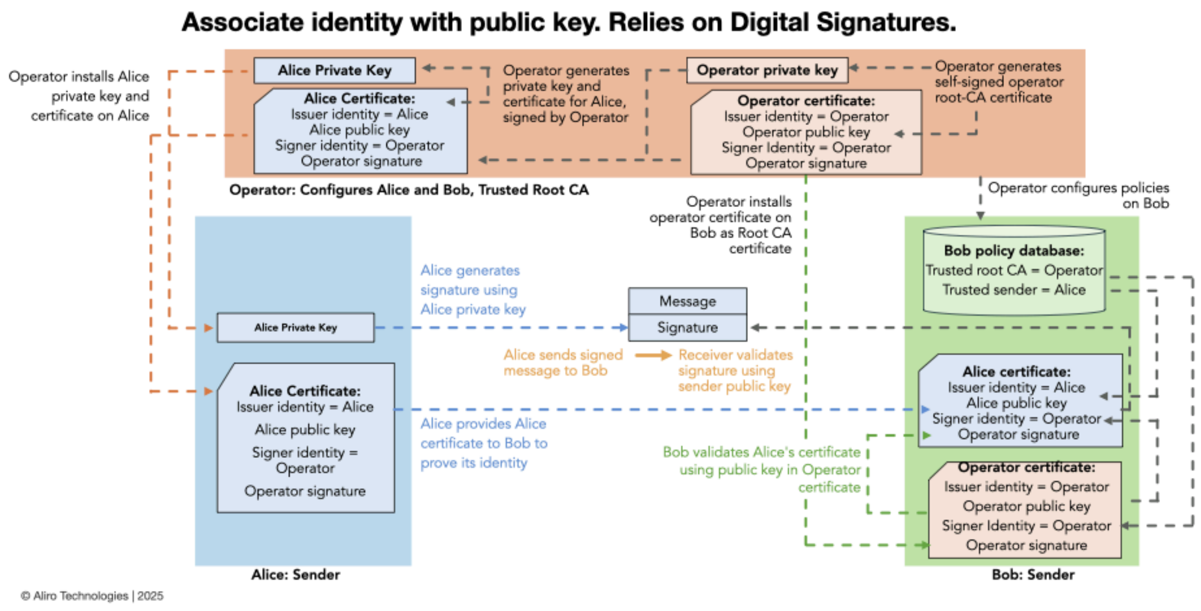
Let's assume that the sender and the receiver have agreed on a public/private key pair. (How this agreement takes place using certificates is discussed later in this white paper.) The private key is only known to the sender. The public key is public information. Anyone, including the receiver, may know the public key. The sender computes a checksum over the sent message and then encrypts this checksum using the private key. The result is a signature which is also known as a hash-based message authentication code or HMAC. The sender sends the signature along with the message itself to the receiver.

The receiver computes the same hash over the received message, decrypts the received signature using the public key, and then the receiver compares these two values. If the decrypted hash matches the locally computed hash, then the receiver knows two things:

1. The message must have been sent by a party who is in possession of the private key. This is part of the authentication procedure.
2. The message was not modified in transit. This is part of integrity verification.

Agreement on a Public / Private Key Pair

How certificates are used in the context of IPsec is a complex process.



Associate identity with public key. Relies on Digital Signatures.
Operator installs Alice private key and certificate on Alice
Alice Private Key
Alice Certificate: Issuer identity = Alice Alice public key Signer identity = Operator Operator signature
Operator generates private key and certificate for Alice, signed by Operator
Operator private key
Operator certificate: Issuer identity = Operator Operator public key Signer Identity = Operator Operator signature
Operator generates self-signed operator root-CA certificate
Operator: Configures Alice and Bob, Trusted Root CA
Operator installs operator certificate on Bob as Root CA certificate
Operator configures policies on Bob
Bob policy database: Trusted root CA = Operator Trusted sender = Alice
Alice generates signature using Alice private key
Message
Signature
Alice sends signed message to Bob
Receiver validates signature using sender public key
Alice provides Alice certificate to Bob to prove its identity
Bob validates Alice's certificate using public key in Operator certificate
Alice certificate: Issuer identity = Alice Alice public key Signer identity = Operator Operator signature
Operator certificate: Issuer identity = Operator Operator public key Signer Identity = Operator Operator signature
Alice: Sender
Bob: Sender
© Aliro Technologies | 2025

At a high level, here is how it works:

- Alice, pictured in blue on the left, uses a digital signature to sign all the messages that she sends to Bob, pictured in green on the right. Alice uses her private key to create these digital signatures.
- Alice also sends her certificates to Bob in the IKE_AUTH exchange, which serves two purposes:
    - The certificate Alice sends to Bob contains Alice's public key, which allows Bob to verify the signature on the messages that are received from Alice.
    - The certificate Alice sends to Bob binds Alice's identity to a specific public key, which allows Bob to authenticate Alice.
- Alice's certificate is signed by a Certificate Authority (CA). In the case of IPsec, the certificate authority is very often the Operator, shown in red at the top, acting as a private certificate authority.
- The operator certificate is installed on Bob as the root of trust, and this is what allows Bob to verify the validity of the certificate received from Alice.

How is IPsec Quantum-Vulnerable?

Which parts of IPsec can be broken by a future quantum computer? There are many commercial companies and academic labs developing a new generation of computers: quantum computers. There are certain things these future quantum computers will be able to do much faster than existing classical computers.

The two quantum algorithms that are most relevant to security are Shor’s algorithm and Grover's algorithm.

Shor's algorithm allows quantum computers to factor large numbers into prime factors and also to compute the discrete logarithm of large numbers very efficiently. The speed up is exponential, which means that a problem that would take a classical supercomputer millions of years to solve could take just minutes, or even seconds, on a quantum computer.

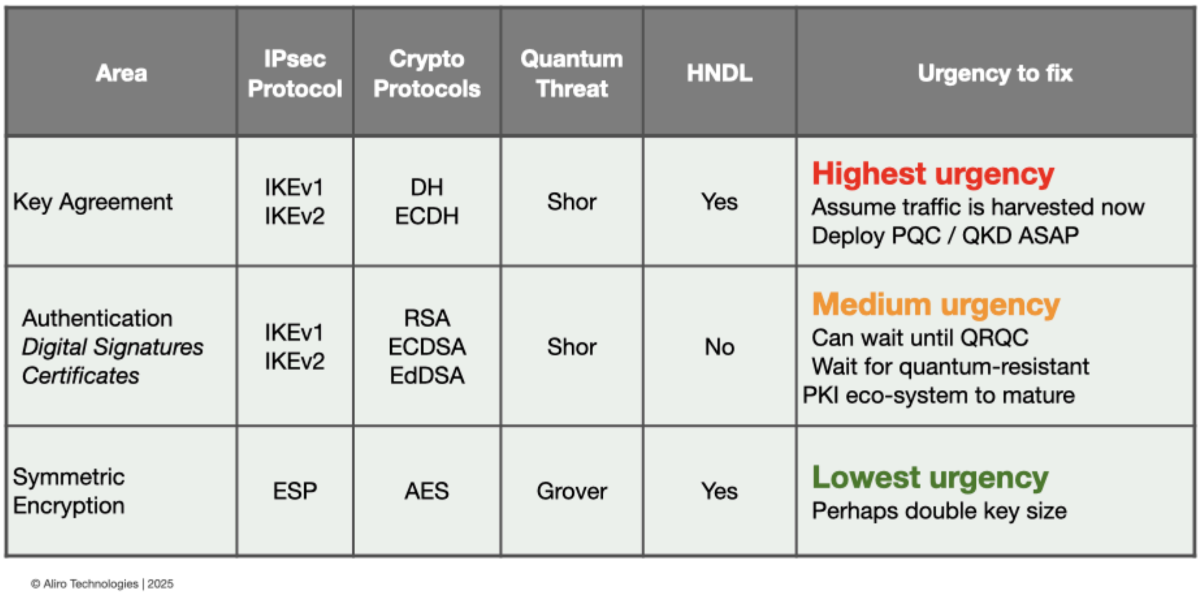
Grover's algorithm allows a quantum computer to search unstructured data very efficiently, as in the case of a brute force key search. The speedup with Grover’s algorithm is quadratic, which is less critical than the logarithmic speedup that Shor’s algorithm is capable of.

A cryptographically relevant quantum computer (CRQC) will be able to run Shor’s algorithm and Grover’s algorithm. Expert estimates on when a CRQC will be implemented vary widely, but the consensus is sometime between 2030 and 2035.

Even though a cryptographically relevant quantum computer has not yet been introduced, it is important to implement post-quantum security now due to harvest now decrypt later (HNDL) attacks. Organizations should assume that governments and cybercriminals are already recording IPsec traffic today, and they will be able to decrypt it several years from now when they have access to a powerful enough quantum computer. This is particularly urgent for financial institutions, medical institutions, critical infrastructure, governmental organizations, defense departments, and high value intellectual property protections. These organizations are likely to be exchanging data today that must remain confidential for more than a few years.

What parts of IPsec are quantum-vulnerable, and how urgent is it to fix it?

There are three parts of IPsec that are quantum-vulnerable.



Area	IPsec Protocol	Crypto Protocols	Quantum Threat	HNDL	Urgency to fix
Key Agreement	IKEv1 IKEv2	DH ECDH	Shor	Yes	**Highest urgency** Assume traffic is harvested now Deploy PQC / QKD ASAP
Authentication *Digital Signatures* *Certificates*	IKEv1 IKEv2	RSA ECDSA EdDSA	Shor	No	**Medium urgency** Can wait until QRQC Wait for quantum-resistant PKI eco-system to mature
Symmetric Encryption	ESP	AES	Grover	Yes	**Lowest urgency** Perhaps double key size

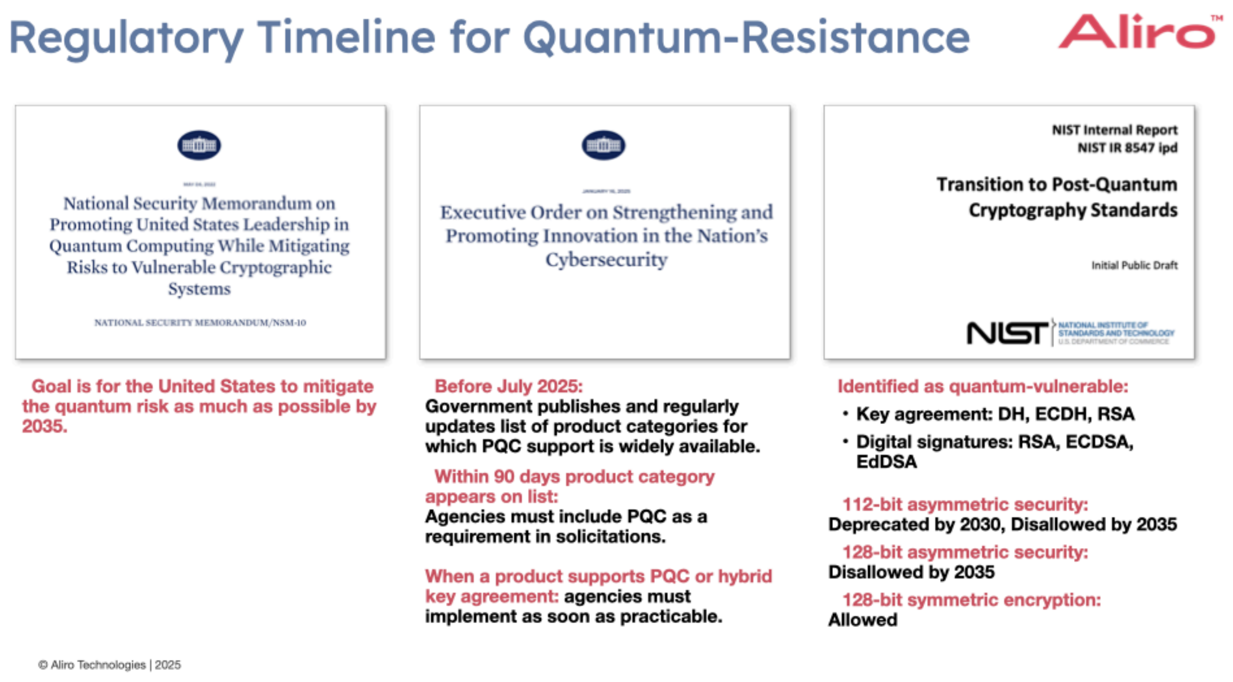
© Aliro Technologies | 2025

1. Key agreement.
2. Certificate based authentication.
3. Symmetric encryption.

IPsec key agreement currently uses Diffie-Hellman or Elliptic Curve Diffie-Hellman, and both of these can be broken by quantum computers using Shor's algorithm. Certificate based authentication uses digital signatures and certificates, which currently rely on RSA, elliptic curve DSA, or Edwards Curve DSA. All of these can also be broken by quantum computers using Shor's algorithm. Finally, all symmetric encryption protocols, including AES and triple DES are at least partially vulnerable to attack by quantum computers due to Grover's algorithm.

The severity of these threats and the urgency for fixing them is different for each of these three categories. Quantum-resistant key agreement is an immediate top priority. Organizations should be doing this as soon as possible. The reason for this high degree of urgency is that this is the part of IPsec that is vulnerable to Harvest Now Decrypt Later (HNDL) attacks. Making key agreements quantum-resistant is an area of standardization that is mostly complete, and where most IPsec vendors have already implemented most or all of the relevant protocol extensions. Quantum-resistant is the second priority. The reason for less urgency here is that unlike key agreements, certificate-based authentication authentication is not vulnerable to HNDL attacks. Any quantum attack on authentication requires a man-in-the-middle attack. It must happen in real time. It cannot happen after the fact. Another reason to wait on addressing

certificate-based authentication is that the standards and the products for quantum resistant public key infrastructure are much more complex and far less mature. Quantum-resilient symmetric encryption is the area of least concern for two reasons. First, because Grover's algorithm provides a quadratic speed up and not an exponential speedup, this vulnerability can be resolved by doubling the key sizes. Although there currently is no standard for AES-512 yet, the other reason for less urgency in this area is that unlike classical brute force attacks, which can easily be parallelized, it appears that Grover's algorithm is not as easily parallelizable. Due to this potential limitation, it’s unclear whether Grover's algorithm will actually be able to reach a quadratic speedup and so it’s unlikely that AES-512 will be required and that AES-256 will be sufficient.
In addition to the technical arguments for implementing quantum-resistant security, there are also some regulatory considerations.



Regulatory Timeline for Quantum-Resistance

Aliro™

National Security Memorandum on Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems

NATIONAL SECURITY MEMORANDUM/NSM-10

Goal is for the United States to mitigate the quantum risk as much as possible by 2035.

Executive Order on Strengthening and Promoting Innovation in the Nation’s Cybersecurity

Before July 2025:
Government publishes and regularly updates list of product categories for which PQC support is widely available.

Within 90 days product category appears on list:
Agencies must include PQC as a requirement in solicitations.

When a product supports PQC or hybrid key agreement: agencies must implement as soon as practicable.

NIST Internal Report
NIST IR 8547 ipd

Transition to Post-Quantum Cryptography Standards

Initial Public Draft

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

Identified as quantum-vulnerable:
- Key agreement: DH, ECDH, RSA
- Digital signatures: RSA, ECDSA, EdDSA

112-bit asymmetric security:
Deprecated by 2030, Disallowed by 2035

128-bit asymmetric security:
Disallowed by 2035

128-bit symmetric encryption:
Allowed

© Aliro Technologies | 2025

The United States government has in the past issued both a memorandum and an executive order requiring government agencies to implement PQC as soon as possible, and to implement and include PQC as procurement requirements under certain circumstances.
NIST has published a draft report, nearing finalization, that deprecates some quantum vulnerable algorithms in 2030 and disallows all quantum vulnerable algorithms in 2035, including all of the asymmetric algorithms that have been discussed here: Diffie-Hellman, RSA, Elliptic Curve Diffie-Hellman, Elliptic Curve DSA, and Edwards Curve DSA. The symmetric encryption protocols, with their current key lengths, continue to be allowed. Other governments in other countries have published similar requirements.

Methods for Implementing Quantum-Safe IPsec

There are several approaches for making IPsec quantum resistant. The first approach is Post Quantum Cryptography, or PQC. Here, the quantum vulnerable algorithms such as Diffie-Hellman and RSA are replaced by newly standardized quantum resistant algorithms such as ML-KEM and ML-DSA. The second approach is Quantum Key Distribution, or QKD. QKD uses quantum physics-based protocols rather than math based protocols to implement quantum resistance.

There are different types of quantum key distribution.
First generation QKD products have been around for more than a decade. They generally use prepare-and-measure protocols, such as BB84. Recently, next-generation QKD systems have been introduced. These are entanglement-based protocols, such as BBM92 or MDI. Entanglement-based QKD systems are more secure, more scalable, and they're more flexible toward a future evolution to a general purpose Quantum Internet. (More information about these different types of QKD can be found in the white paper The Fundamentals of Quantum Secure Communication.) As far as quantum-safe IPsec is concerned, it doesn't matter which of these QKD technologies is deployed. Each of these QKD technologies prepare keys and deliver those keys to IPsec using the same standard key delivery interface that is discussed in detail later in this white paper.

An important method to consider is hybrid quantum resistance, where multiple crypto algorithms are combined. For example, combining traditional Diffie-Hellman with one or more PQC algorithms such as ML-KEM, alongside QKD implementation. One advantage of this hybrid approach is defense-in-depth: the security of the entire combined system is as strong as the strongest contributing security mechanism. In other words, an attacker would have to break each crypto algorithm individually in order to break the overall security of the entire system. The second advantage is that it helps with regulatory compliance. Many security standards and regulations have not yet been updated for the post quantum world, and they still require the use of quantum-vulnerable algorithms such as Diffie-Hellman and RSA. By including them in the hybrid mix, it's possible to maintain standards compliance and regulations compliance. The disadvantage to this hybrid approach is that it requires additional resources such as processing, memory, and bandwidth in order to generate and verify multiple keys and signatures. It also adds complexity to standardization and implementation. This is not particularly complex in the case of IPsec, but more complex in the case of TLS and PKI.

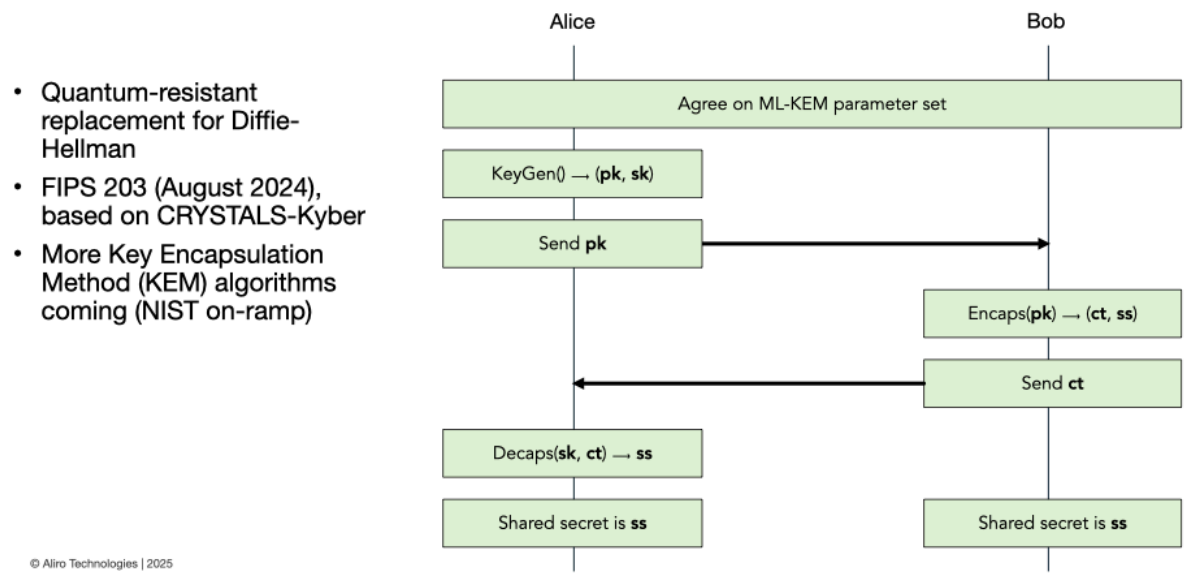
Standards for adding PQC to IPsec

This section focuses on the extensions to IPsec that enable Post Quantum Cryptography (PQC) for key exchange, ensuring that encrypted communications remain secure against quantum-enabled threats. However, this discussion is limited to securing the IPsec key exchange process and does not cover IPsec authentication.

Making IPsec authentication quantum-safe is a separate challenge that closely aligns with post-quantum advancements in Public Key Infrastructure (PKI) and digital certificates. While NIST has already published the first PQC standards for authentication, ongoing efforts within the IETF and commercial sectors are still refining how to integrate quantum-safe authentication into control-plane protocols. For now, we will explore how PQC-based key exchange is being standardized and implemented within IPsec, ensuring that organizations can proactively defend their encrypted communications against emerging quantum threats.

Module-Lattice Key Encapsulation Method (ML-KEM)

In August of 2024, NIST published FIPS 203, the standard for the first quantum-safe algorithm to replace quantum-vulnerable Diffie-Hellman. The name of the new algorithm is Module Lattice Key Encapsulation Method, or ML-KEM. ML-KEM was previously referred to as CRYSTALS-Kyber, the code name that was used during the standardization process.


- Quantum-resistant replacement for Diffie-Hellman
- FIPS 203 (August 2024), based on CRYSTALS-Kyber
- More Key Encapsulation Method (KEM) algorithms coming (NIST on-ramp)

At a high level, ML-KEM works as follows:

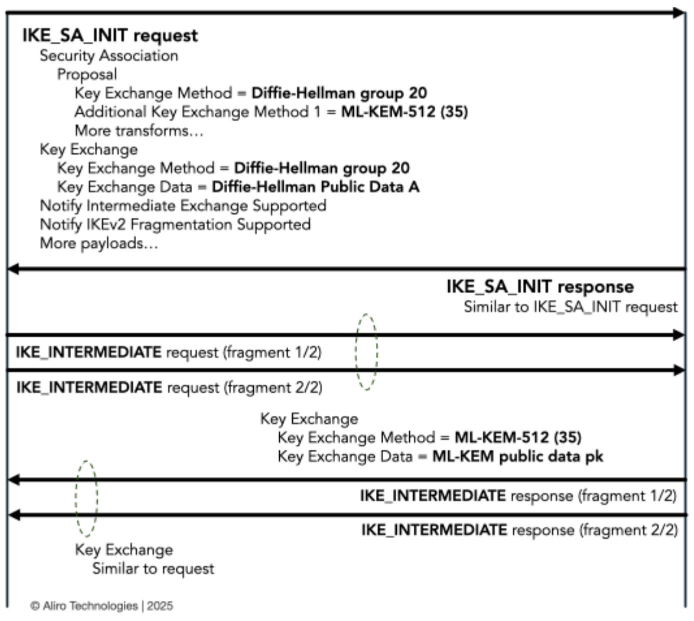
- Alice, pictured on the left in the above image, performs a KeyGen operation. This generates a public key (pk) and a secret key (sk).
- Alice then sends the public key to Bob and keeps the secret key.
- Bob performs an Encaps operation, consuming the public key that was received from Alice and generating a clear text (ct) and a shared secret (ss).
- Bob keeps the shared secret (ss) and sends the clear text (ct) back to Alice.

- Lastly, Alice uses the secret key (the one Alice kept) and the clear text (received from Bob) to perform a Decaps operation, and generate the same shared secret (ss) as Bob.

These operations are based on a mathematical problem which is known as the Module Learning with Errors problem, and is different from Diffie-Hellman, which uses factorization. There's a strong belief that it's computationally infeasible for an eavesdropper to reverse engineer the shared secrets from these publicly communicated values, even if the eavesdropper has a quantum computer and even if the eavesdropper can run Shor's algorithm.

IKEv2 PQC Key Exchange

Applying this PQC methodology to the same IKEv2 exchange discussed previously shows how it has been extended to support Post Quantum Cryptography. The relevant RFCs for each step are listed here on the right side of the image below.



IKE_SA_INIT request
Security Association
Proposal
Key Exchange Method = **Diffie-Hellman group 20**
Additional Key Exchange Method 1 = **ML-KEM-512 (35)**
More transforms…
Key Exchange
Key Exchange Method = **Diffie-Hellman group 20**
Key Exchange Data = **Diffie-Hellman Public Data A**
Notify Intermediate Exchange Supported
Notify IKEv2 Fragmentation Supported
More payloads…
IKE_SA_INIT response
Similar to IKE_SA_INIT request
IKE_INTERMEDIATE request (fragment 1/2)
IKE_INTERMEDIATE request (fragment 2/2)
Key Exchange
Key Exchange Method = **ML-KEM-512 (35)**
Key Exchange Data = **ML-KEM public data pk**
IKE_INTERMEDIATE response (fragment 1/2)
IKE_INTERMEDIATE response (fragment 2/2)
Key Exchange
Similar to request
© Aliro Technologies | 2025

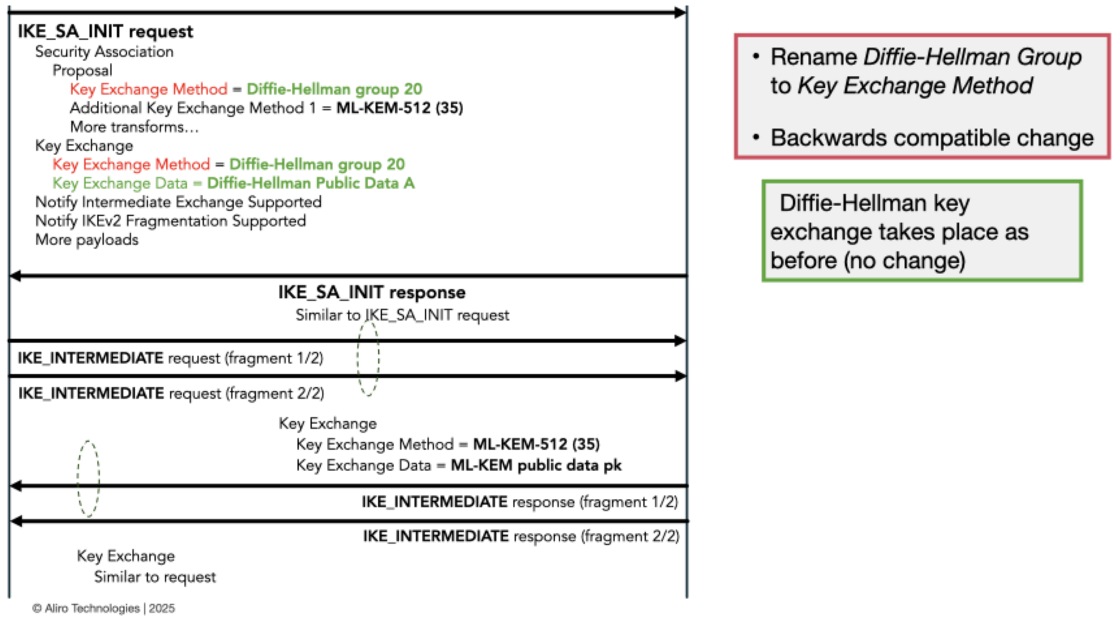
draft-kampanakis-ml-kem-ikev2: Post-quantum Hybrid Key Exchange with ML-KEM in the Internet Key Exchange Protocol Version 2 (IKEv2))

RFC9242: Intermediate Exchange in the Internet Key Exchange Protocol Version 2 (IKEv2)

RFC9470: Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)

RFC7383: Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation

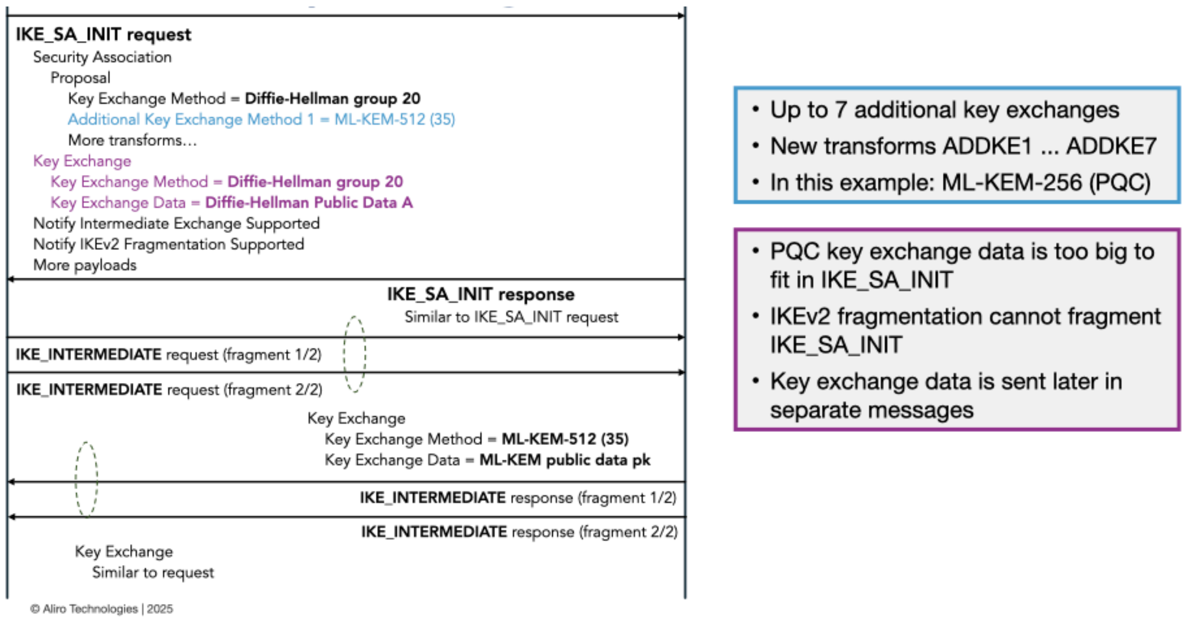
The first change, highlighted in red below, is that the Diffie-Hellman group payload has been generalized and has been renamed to Key Exchange Method.


IKE_SA_INIT request
Security Association
Proposal
Key Exchange Method = Diffie-Hellman group 20
Additional Key Exchange Method 1 = ML-KEM-512 (35)
More transforms…
Key Exchange
Key Exchange Method = Diffie-Hellman group 20
Key Exchange Data = Diffie-Hellman Public Data A
Notify Intermediate Exchange Supported
Notify IKEv2 Fragmentation Supported
More payloads
IKE_SA_INIT response
Similar to IKE_SA_INIT request
IKE_INTERMEDIATE request (fragment 1/2)
IKE_INTERMEDIATE request (fragment 2/2)
Key Exchange
Key Exchange Method = ML-KEM-512 (35)
Key Exchange Data = ML-KEM public data pk
IKE_INTERMEDIATE response (fragment 1/2)
IKE_INTERMEDIATE response (fragment 2/2)
Key Exchange
Similar to request
© Aliro Technologies | 2025
• Rename Diffie-Hellman Group to Key Exchange Method
• Backwards compatible change
Diffie-Hellman key exchange takes place as before (no change)

This is a backwards compatible change: it's just renaming the element. When a Diffie-Hellman group is negotiated, the binary representation of the payload doesn't change at all. IPsec still does a traditional Diffie-Hellman key exchange, as highlighted in green above, even if PQC is enabled. (The reason for this will become clear later.)

When PQC is enabled, IPsec typically does additional rounds of PQC key exchanges in addition to the traditional Diffie-Hellman exchange. In other words, IPsec typically does a hybrid key exchange when PQC is enabled.

There are new transform types for these additional rounds of key agreements.

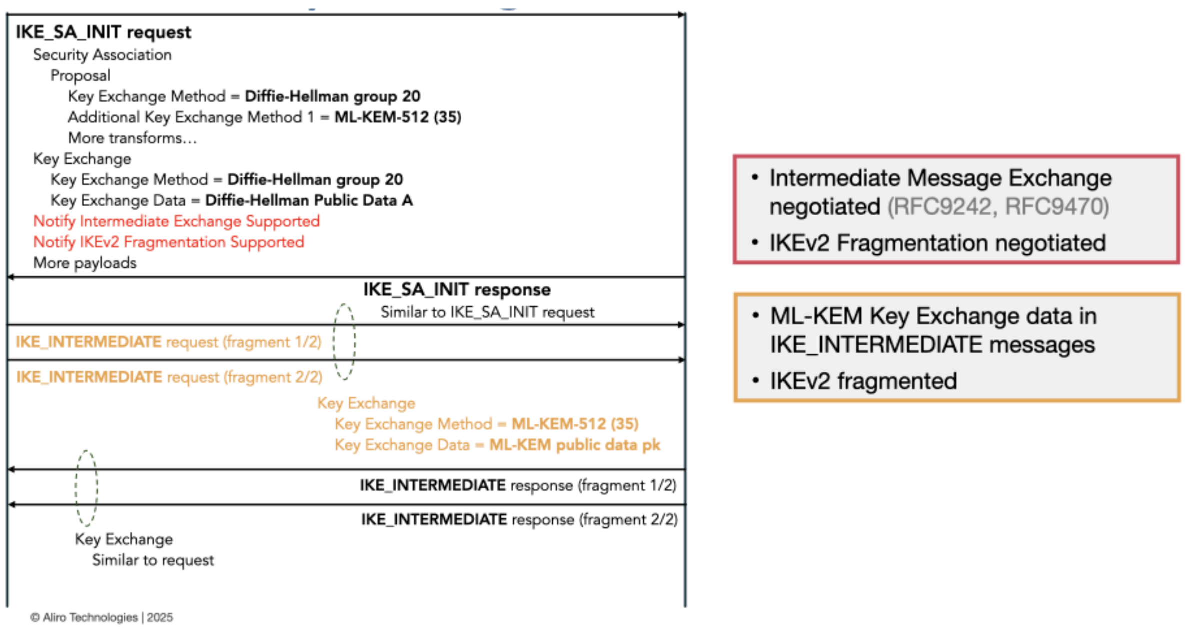

IKE_SA_INIT request
Security Association
Proposal
Key Exchange Method = Diffie-Hellman group 20
Additional Key Exchange Method 1 = ML-KEM-512 (35)
More transforms…
Key Exchange
Key Exchange Method = Diffie-Hellman group 20
Key Exchange Data = Diffie-Hellman Public Data A
Notify Intermediate Exchange Supported
Notify IKEv2 Fragmentation Supported
More payloads
IKE_SA_INIT response
Similar to IKE_SA_INIT request
IKE_INTERMEDIATE request (fragment 1/2)
IKE_INTERMEDIATE request (fragment 2/2)
Key Exchange
Key Exchange Method = ML-KEM-512 (35)
Key Exchange Data = ML-KEM public data pk
IKE_INTERMEDIATE response (fragment 1/2)
IKE_INTERMEDIATE response (fragment 2/2)
Key Exchange
Similar to request
© Aliro Technologies | 2025

- Up to 7 additional key exchanges
- New transforms ADDKE1 ... ADDKE7
- In this example: ML-KEM-256 (PQC)

- PQC key exchange data is too big to fit in IKE_SA_INIT
- IKEv2 fragmentation cannot fragment IKE_SA_INIT
- Key exchange data is sent later in separate messages

In the example above, there is one additional key exchange protocol being negotiated, ML-KEM 256, which is highlighted in blue. Note that there is no corresponding PQC key exchange payload in the IKE_SA_INIT message. There is only a key exchange payload for the traditional Diffie-Hellman exchange, highlighted in purple. The reason for the additional PQC key exchange payloads not being in the IKE_SA_INIT message is that PQC public keys are much larger than traditional keys. In fact, they are so large that they don't fit in UDP packets without fragmentation. IPsec has its own fragmentation mechanism, but IPsec does not allow the first IKE_SA_INIT message to be fragmented. For this reason, the key exchange payloads for the additional PQC keys are moved from this message to a separate, follow up message.

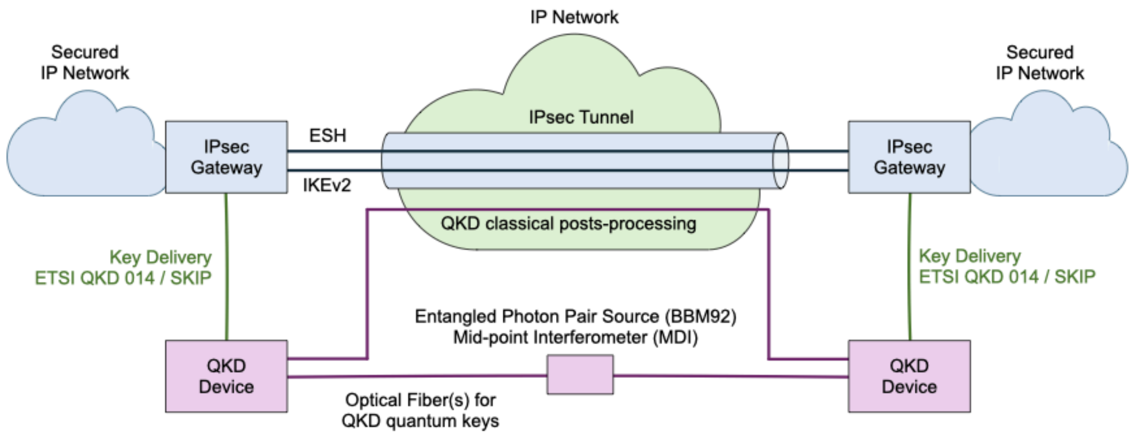
There are two new IPsec RFCs that have been introduced to allow for these additional rounds of PQC exchanges after the initial round of Diffie-Hellman key exchanges. RFC9242 introduces a new message type which is called IKE Intermediate, pictured below. RFC9470 describes how this new message type can be used to do multiple rounds of hybrid key exchange.


IKE_SA_INIT request
Security Association
Proposal
Key Exchange Method = **Diffie-Hellman group 20**
Additional Key Exchange Method 1 = **ML-KEM-512 (35)**
More transforms…
Key Exchange
Key Exchange Method = **Diffie-Hellman group 20**
Key Exchange Data = **Diffie-Hellman Public Data A**
Notify Intermediate Exchange Supported
Notify IKEv2 Fragmentation Supported
More payloads
IKE_SA_INIT response
Similar to IKE_SA_INIT request
IKE_INTERMEDIATE request (fragment 1/2)
IKE_INTERMEDIATE request (fragment 2/2)
Key Exchange
Key Exchange Method = **ML-KEM-512 (35)**
Key Exchange Data = **ML-KEM public data pk**
IKE_INTERMEDIATE response (fragment 1/2)
IKE_INTERMEDIATE response (fragment 2/2)
Key Exchange
Similar to request
© Aliro Technologies | 2025
• Intermediate Message Exchange negotiated (RFC9242, RFC9470)
• IKEv2 Fragmentation negotiated
• ML-KEM Key Exchange data in IKE_INTERMEDIATE messages
• IKEv2 fragmented

In this example, there is one additional IKE Intermediate exchange for the extra single round of PQC key exchange using ML-KEM. The key exchange payload in this extra intermediate message is used to carry Alice's public key and Bob's clear text. Since this public key is too large to fit in a single UDP packet, this IKE Intermediate message is IKEv2 fragmented. The IKE_SA_INIT message contains notifications to negotiate the use of this new intermediate key exchange feature and of the existing fragmentation feature, both of which are needed to make this work.

Standards for adding QKD to IPsec

The image below shows how IPsec is deployed in combination with Quantum Key Distribution (QKD).

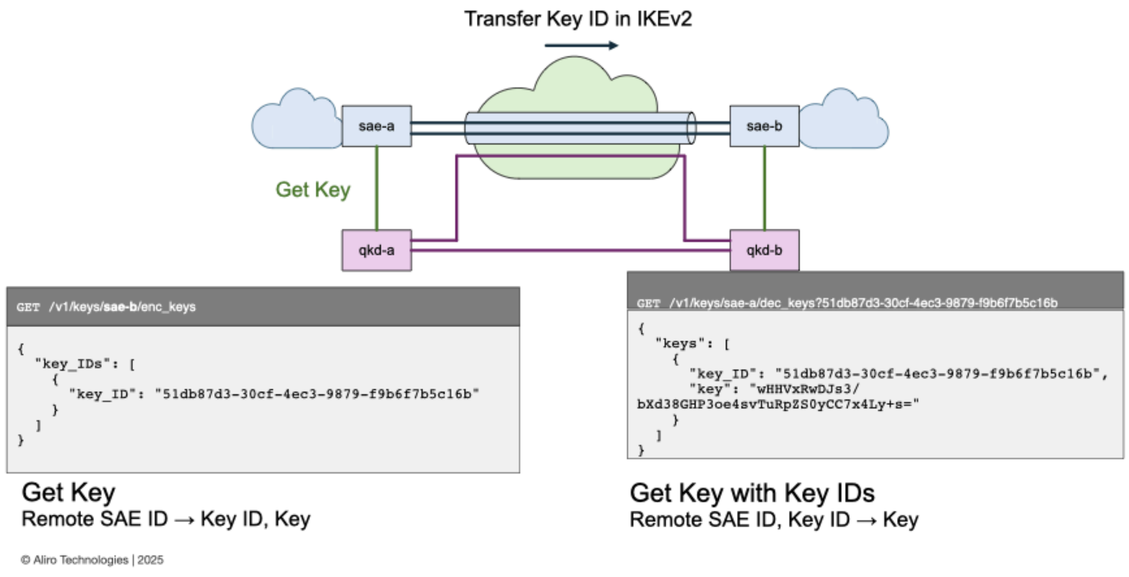

IP Network
Secured IP Network
Secured IP Network
IPsec Tunnel
IPsec Gateway
IPsec Gateway
ESH
IKEv2
QKD classical posts-processing
Key Delivery ETSI QKD 014 / SKIP
Key Delivery ETSI QKD 014 / SKIP
Entangled Photon Pair Source (BBM92)
Mid-point Interferometer (MDI)
QKD Device
QKD Device
Optical Fiber(s) for QKD quantum keys

There are two IPsec gateways with an IPsec tunnel in between. There are also two QKD devices, one on each side, which are responsible for producing the quantum-safe keys. The two QKD devices use the quantum properties of individual photons to exchange keys with each other over dedicated fibers, as shown in purple in the image above.

Key Delivery Protocols: ETSI QKD 014 / SKIP

Once the QKD devices have produced a key, they hand this key over to the IPsec gateway using a key delivery interface. There are two different protocols for the key delivery interface: ETSI QKD 014 and SKIP.

ETSI QKD 014 was standardized by the European Telecommunication Standardization Institute (ETSI). The other protocol is the Secure Key Integration Protocol, or SKIP, which was defined by Cisco and has recently been documented in a public IETF draft. Most QKD devices support both ETSI QKD and SKIP, and most IPsec encryptors use ETSI QKD except for Cisco encryptors that typically use SKIP. Both protocols are similar in how they operate. Note that both ETSI QKD 014 and SKIP are completely agnostic when it comes to which QKD protocol is being used. ETSI QKD 014 and SKIP are mechanisms for handing the produced key to the encryptor, regardless of the QKD protocol used to generate the key. Below is an example of a pair of QKD devices delivering a pair of symmetric encryption keys to a pair of IPsec firewalls.



```
GET /v1/keys/sae-b/enc_keys
{
  "key_IDs": [
    {
      "key_ID": "51db87d3-30cf-4ec3-9879-f9b6f7b5c16b"
    }
  ]
}
```

Get Key
Remote SAE ID → Key ID, Key

```
GET /v1/keys/sae-a/dec_keys?51db87d3-30cf-4ec3-9879-f9b6f7b5c16b
{
  "keys": [
    {
      "key_ID": "51db87d3-30cf-4ec3-9879-f9b6f7b5c16b",
      "key": "wHHVxRwDJs3/
bXd38GHP3oe4svTuRpZS0yCC7x4Ly+s="
    }
  ]
}
```

Get Key with Key IDs
Remote SAE ID, Key ID → Key

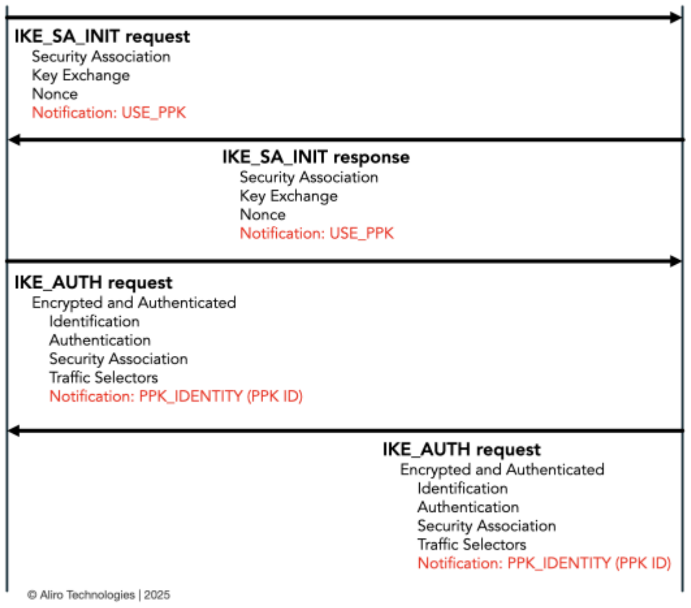
This particular example is using ETSI QKD 014, but an example using SKIP would look nearly identical. Both ETSI QKD 014 and SKIP are REST interfaces that run over https. The flow of messages is as follows:

- The IPsec gateway, pictured on the left in the image above, first requests the key from the co-located QKD device using what's known as a Get Key request.
- The Get Key request returns the actual key value, which is secret, and a key identifier, which is public.
- IPsec transfers this public key ID to the remote IPsec gateway using an IKEv2 message. (Details of this process are discussed below.)
- The IPsec gateway on the right requests the same key from its co-located QKD device using a Get Key with Key IDs request.
- The encryptor provides the key ID in the request and receives the key value.

At this point, the two IPsec gateways have the same symmetric shared key and they start encrypting the traffic in the IPsec tunnel.

IKEv2 Post-quantum Pre-shared Keys (PPK)

The image below shows how the IKEv2 protocol has been extended to transfer the public key identifier from one IPsec gateway to the other.

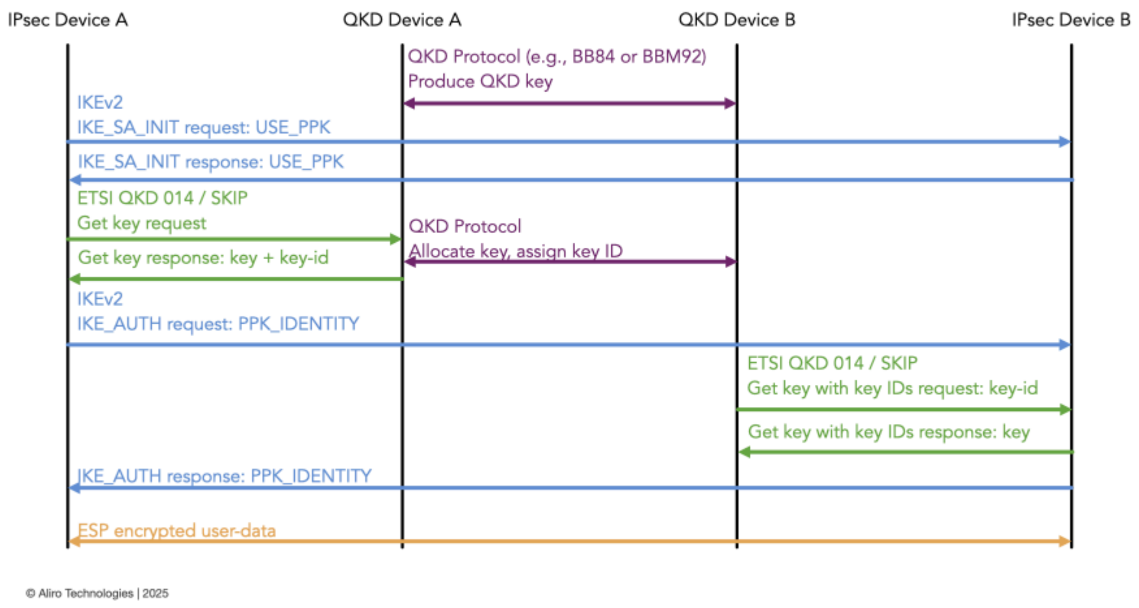


PPK is mixed in during initial IKE SA establishment

```
SKEYSEED = prf(Ni | Nr, g^ir)
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' |
SK_pr'}
   = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr)
SK_d = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')
```

These extensions are standardized in RFC8784, which defines a new payload called the Post-quantum Preshared Key or PPK. The term “post-quantum preshared” might seem confusing; it's called “preshared” because RFC8784 was originally envisioned to use manually configured shared quantum-safe keys. It was only later that RFC8784 was generalized to be used for quantum keys dynamically produced by QKD devices. The protocol is simple: in the IKE_SA_INIT exchange there is a USE_PPK notification to negotiate the use of this extension. Then, in the IKE_AUTH exchange, the actual key identifier is transferred using the PPK_IDENTITY notification. RFC8784 also describes how to mix this QKD produced key with the rest of the key material, and this is shown in the box in gray on the right in the image above.

Pictured below is an end-to-end example of how all the various protocols work together to use QKD in the context of IPsec.


IPsec Device A
QKD Device A
QKD Device B
IPsec Device B
QKD Protocol (e.g., BB84 or BBM92)
Produce QKD key
IKEv2
IKE_SA_INIT request: USE_PPK
IKE_SA_INIT response: USE_PPK
ETSI QKD 014 / SKIP
Get key request
QKD Protocol
Allocate key, assign key ID
Get key response: key + key-id
IKEv2
IKE_AUTH request: PPK_IDENTITY
ETSI QKD 014 / SKIP
Get key with key IDs request: key-id
Get key with key IDs response: key
IKE_AUTH response: PPK_IDENTITY
ESP encrypted user-data
© Aliro Technologies | 2025

- The purple arrows represent the quantum and classical post-processing protocols between the QKD devices. There are no standards for this yet, so these purple arrows are currently typically proprietary protocols from the QKD vendor.
- The blue arrows represent the IKEv2 control plane messages for the IPsec tunnel.
- The green arrows represent the ETSI QKD 014 or the SKIP key delivery protocol between the QKD device and the IPsec gateway.
- The yellow arrows represent the ESP data plane protocol for the IPsec tunnel that is encrypted using the QKD data.

The sequence of events is as follows:

- The QKD devices are constantly producing key material.
- At some point in time the two IPsec gateways decide that they want to establish an IPsec tunnel.
- In the initial IKE_SA_INIT exchange they negotiate the use of PPK (Post-quantum Preshared Keys) and this roughly translates into requesting to use QKD.
- The initiator IPsec gateway requests the key from its co-located QKD device using a Get Key API call.
- Behind the scenes, the two QKD devices collaborate with each other to assign QKD material to the requested key and to assign a key identifier to it.
- Then the key value and the key identifier are returned to the initial IPsec gateway in the response to the Get Key request.

- The initiator IPsec Gateway transfers this key ID to the responder IPsec gateway into PPK identity notification, which is in the IKE_AUTH message.
- The IPsec gateway on the responder side then invokes the Get Key with Key IDs API call on its co-located QKD device, providing the public key ID that it received in the IKE_AUTH message. It gets back the secret key value.
- At this point, both IPsec gateways have the same shared secret encryption key that was produced by QKD.

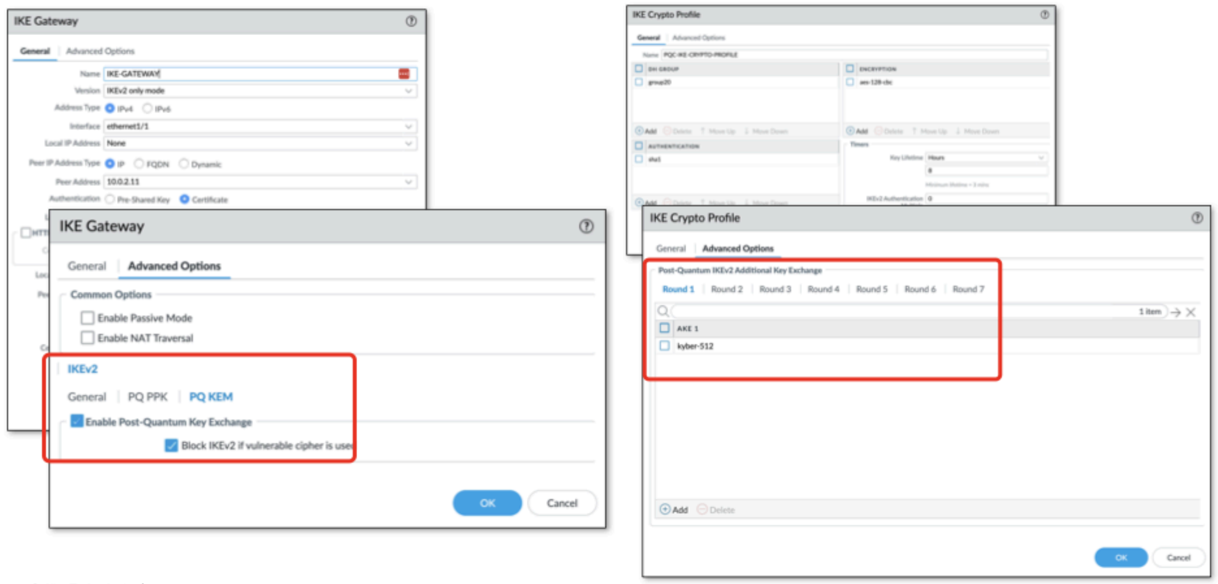
The IPsec gateways typically mix this QKD key with a traditional Diffie-Hellman key, and sometimes also with one or more PQC keys. A final resulting mixed key is used to encrypt the ESP data plane.

Real-World Examples

The extensions in IPsec for PQC and QKD that have been described so far are somewhat complex, but fortunately these complexities occur behind the scenes and are not visible to network operators or users. In practice, it's simple and straightforward for operators to configure and monitor PQC and QKD for their IPsec networks, as will be shown in the examples here.

Example 1: Configure PQC

Below is the screenshot of the management web interface of one widely used IPsec gateway product.


IKE Gateway — General

Name: IKE-GATEWAY
Version: IKEv2 only mode
Address Type: IPv4
Interface: ethernet1/1
Local IP Address: None
Peer IP Address Type: IP
Peer Address: 10.0.2.11
Authentication: Certificate

IKE Gateway — Advanced Options

Common Options
[ ] Enable Passive Mode
[ ] Enable NAT Traversal

IKEv2 — PQ KEM
[✓] Enable Post-Quantum Key Exchange
[✓] Block IKEv2 if vulnerable cipher is use

IKE Crypto Profile — General

Name: PQC-IKE-CRYPTO-PROFILE
DH GROUP: group20
ENCRYPTION: aes-128-cbc
AUTHENTICATION: sha1
Key Lifetime: Hours, 8
IKEv2 Authentication: 0

IKE Crypto Profile — Advanced Options

Post-Quantum IKEv2 Additional Key Exchange — Round 1
AKE 1: kyber-512

© Aliro Technologies | 2025

It shows the screens for configuring an IKE gateway and an IKE crypto profile. While these screens contain many configuration fields, the vast majority of these fields are things that must be configured anyway as part of the normal workflow for configuring any IPsec tunnel.

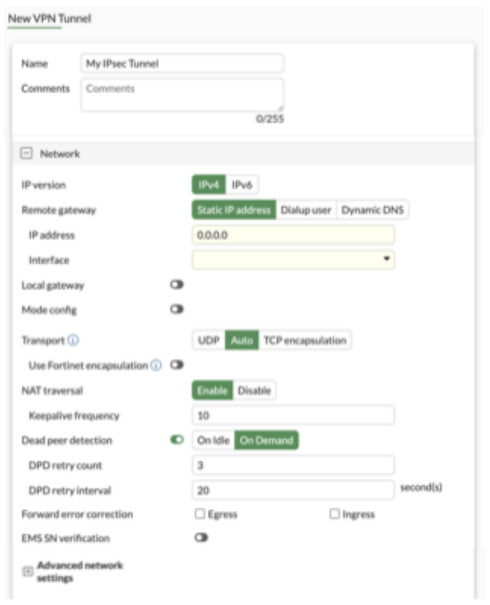
The new options for adding PQC are marked in red above. Enabling PQC for your IPsec tunnel is very straightforward and only involves a couple of new configuration fields.

1. Check the “enable post quantum key exchange” checkbox.
2. Select one or multiple post quantum key exchange methods.

The example shows the selection of Kyber-512. This is all that is required in order to turn on PQC in the network using software that is commercially available today.

Example 2: Configure PQC

Below is a screenshot of the management web interface of another widely-used IPsec product from a different vendor.



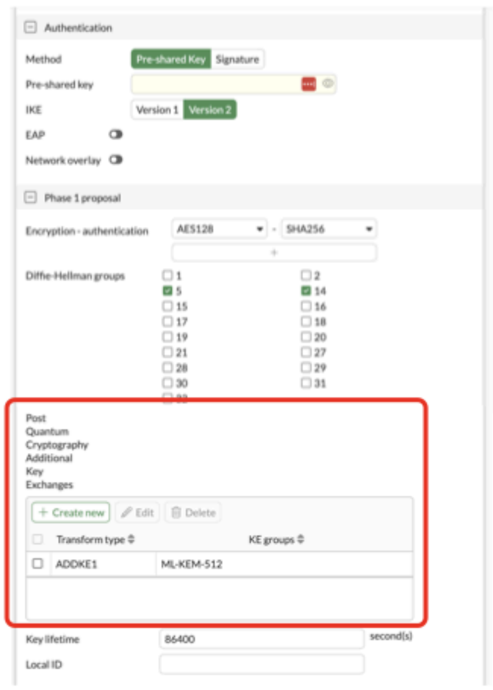
New VPN Tunnel

Name	My IPsec Tunnel
Comments	Comments 0/255

Network

IP version	[✓ IPv4] IPv6
Remote gateway	[✓ Static IP address] Dialup user Dynamic DNS
IP address	0.0.0.0
Interface	
Local gateway	
Mode config	
Transport	UDP [✓ Auto] TCP encapsulation
Use Fortinet encapsulation	
NAT traversal	[✓ Enable] Disable
Keepalive frequency	10
Dead peer detection	On Idle [✓ On Demand]
DPD retry count	3
DPD retry interval	20 second(s)
Forward error correction	☐ Egress ☐ Ingress
EMS SN verification	

Advanced network settings



Authentication

Method	[✓ Pre-shared Key] Signature
Pre-shared key	
IKE	Version 1 [✓ Version 2]
EAP	
Network overlay	

Phase 1 proposal

Encryption - authentication	AES128 - SHA256
Diffie-Hellman groups	☐ 1 ☐ 2 ✓ 5 ✓ 14 ☐ 15 ☐ 16 ☐ 17 ☐ 18 ☐ 19 ☐ 20 ☐ 21 ☐ 27 ☐ 28 ☐ 29 ☐ 30 ☐ 31 ☐ 32

Post Quantum Cryptography Additional Key Exchanges

+ Create new Edit Delete

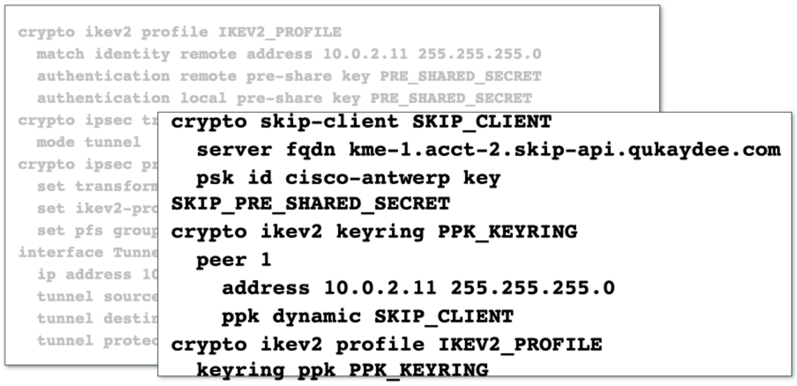
	Transform type	KE groups
☐	ADDKE1	ML-KEM-512

Key lifetime	86400 second(s)
Local ID	

The screen for creating an IPsec tunnel pictured above shows that the only step required is selecting one of the PQC key exchange methods. In this example, ML-KEM 512 is selected. This is the only step for turning on PQC in the network using this particular vendor’s software.

Example 3: Configure QKD using CLI

Below is an example of how to configure QKD using SKIP. In this case, the example shows how to configure using QKD with the command line interface (CLI) instead of the graphical user interface.



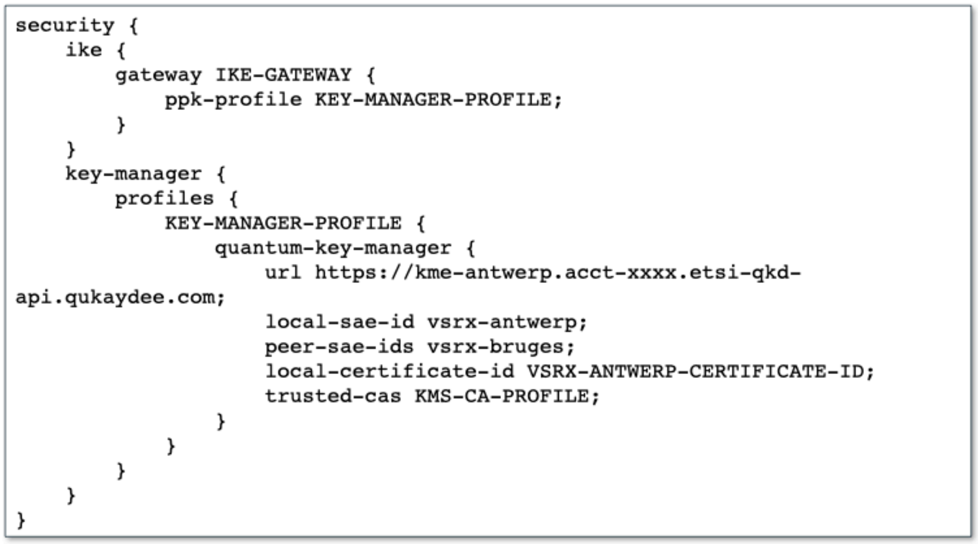
```
crypto ikev2 profile IKEV2_PROFILE
  match identity remote address 10.0.2.11 255.255.255.0
  authentication remote pre-share key PRE_SHARED_SECRET
  authentication local pre-share key PRE_SHARED_SECRET
crypto ipsec tr
  mode tunnel
crypto ipsec pr
  set transform
  set ikev2-pro
  set pfs group
interface Tunne
  ip address 10
  tunnel source
  tunnel destin
  tunnel protec
```

```
crypto skip-client SKIP_CLIENT
  server fqdn kme-1.acct-2.skip-api.qukaydee.com
  psk id cisco-antwerp key
SKIP_PRE_SHARED_SECRET
crypto ikev2 keyring PPK_KEYRING
  peer 1
    address 10.0.2.11 255.255.255.0
    ppk dynamic SKIP_CLIENT
crypto ikev2 profile IKEV2_PROFILE
  keyring ppk PPK_KEYRING
```

© Aliro Technologies | 2025

The configuration (pictured in the background in gray above) is the configuration required for any IPsec tunnel. In black (at the front) is the additional configuration for enabling QKD. Essentially this process is as simple as indicating the need to use QKD and entering some information to identify how to reach the QKD device to get the key.

Example 4: Configure QKD using CLI



```
security {
    ike {
        gateway IKE-GATEWAY {
            ppk-profile KEY-MANAGER-PROFILE;
        }
    }
    key-manager {
        profiles {
            KEY-MANAGER-PROFILE {
                quantum-key-manager {
                    url https://kme-antwerp.acct-xxxx.etsi-qkd-
api.qukaydee.com;
                    local-sae-id vsrx-antwerp;
                    peer-sae-ids vsrx-bruges;
                    local-certificate-id VSRX-ANTWERP-CERTIFICATE-ID;
                    trusted-cas KMS-CA-PROFILE;
                }
            }
        }
    }
}
```

This is another example of using QKD. In this case, ETSI QKD 014 is used for the key delivery interface. Shown above is the configuration needed in addition to configuring the IPsec tunnel. A majority of this configuration is about where to get the key and how to authenticate your identity to the key delivery QKD device.

Conclusion

If you are using IPsec in your organization, it is important to begin planning for making your IPsec deployment quantum-safe as soon as possible. Specifically, you should learn about the technology and begin trials now to understand the impact on your organization. Securing IPsec key exchange is the most urgent migration needed to protect data-in-transit from today's threats as well as future threats. Not only is data already at risk because of harvest now decrypt later attacks, but there are also regulatory requirements demanding these vulnerabilities be addressed in the near-term.

Deploying PQC is the baseline preparation for a post-quantum world, but if you are using IPsec to protect long-term confidential information, you should also consider deploying QKD in addition to PQC, as an additional layer of security. Applications that benefit from this additional layer of security include critical infrastructure, medical applications, financial applications, high value intellectual property protections, government data, defense security, and anything that requires long-term protection.

If you choose to deploy QKD, you have a choice of a variety of QKD protocols. Using an entanglement-based QKD protocol offers a higher level of security, better scalability, and more flexibility for deploying a general purpose quantum network that can connect a wide variety of quantum devices for applications beyond secure key distribution.

For further information, please see How to configure an IPsec tunnel using PQC keys and How to configure an IPsec tunnel using QKD keys.

The Future is Entanglement-based Quantum Networks

As we look to the future of secure communications, entanglement-based quantum networks are at the forefront. Building these networks requires specialized software and hardware, such as beam splitters.

Entanglement-based quantum networks are being built today by a variety of organizations for a variety of use cases – benefiting organizations internally, as well as providing great value to an organization's customers. Telecommunications companies, national research labs, and systems integrators are just a few examples of the organizations Aliro is helping to leverage the capabilities of quantum secure communications.

Aliro is uniquely positioned to help you build your quantum network. The steps you can take to ensure your organization is meeting the challenges and leveraging the benefits of the quantum revolution are part of a clear, unified solution already at work in networks like the EPB Quantum Network℠ powered by Qubitekk in Chattanooga, Tennessee.

AliroNet™, the world's first full-stack entanglement-based network solution, consists of the software and services necessary to ensure customers will fully meet their advanced secure networking goals. Each component within AliroNet™ is built from the ground up to be compatible and optimal with entanglement-based networks of any scale and architecture. AliroNet™ is used to simulate, design, run, and manage quantum networks as well as test, verify, and optimize quantum hardware for network performance. AliroNet™ leverages the expertise of Aliro personnel in order to ensure that customers get the most value out of the software and their investment.

Depending on where customers are in their quantum networking journeys, AliroNet™ is available in three modes that create a clear path toward building full-scale entanglement-based secure networks: (1) Emulation Mode, for emulating, designing, and validating entanglement-based quantum networks, (2) Pilot Mode for implementing a small-scale entanglement-based quantum network testbed, and (3) Deployment Mode for scaling entanglement-based quantum networks and integrating end-to-end applications. AliroNet™ has been developed by a team of world-class experts.

To get started on your Quantum Networking journey, reach out to the Aliro team for additional information on how AliroNet™ can enable secure communications.