ETSI TR 103 617 V1.1.1 (2018-09)

Quantum-Safe Virtual Private Networks

Reference

DTR/CYBER-QSC-009

Keywords

quantum cryptography, security

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Recent research in the field of quantum computing has brought about a credible threat to the current state-of-the-art for protecting electronic information [i.1]. The current data protection mechanisms that typically comprise cryptographic systems rely on computational hardness as a means to protect sensitive data. This is to say that there are cryptographic problems that are difficult or impossible to solve using conventional computing.

Because of recent advances in quantum computing, the quantum computer presents a serious challenge to widely used current cryptographic techniques and assumptions. This is because the quantum computer tends to excel at certain classes of problems. Among these problem classes are:

1) the integer factorization problem, which is used by the Rivest Shamir Adleman (RSA) cryptographic system; and

2) the discrete logarithms problem, which is used by Elliptic Curve Cryptography (ECC).

Both RSA and ECC are common public-key cryptographic techniques that are used to secure much of the interchange of information over the Internet as of 2017. While the integer factorization and discrete logs problems are difficult or practically impossible to solve using a conventional computer, they become fairly trivial for a quantum computer.

Academia, industry and governments have all made large investments in building a universal quantum computer powerful enough to break currently used public-key algorithms. Therefore, new solutions based on hard problems that cannot be efficiently solved by algorithms running on a quantum computer, such as Shor's algorithm, are needed to secure the existing cryptographic protocols [i.2]. Once the appropriate replacements for currently used cryptographic primitives are selected, these protocols can be updated. There is nevertheless an immediate harvest and decrypt threat from a quantum-capable adversary.

The deployment of Virtual Private Networks (VPNs) is a common choice for governments and enterprises to securely communicate between sites or to connect employees with offices.

Figure 1 describes how the harvest and decrypt attack would work against a VPN session. Each VPN session consists of two stages:

1) the handshake; and

2) data exchange between the two parties.

During the handshake stage, the peers are authenticated, and the symmetric keys are established. Once that has been completed, the peers can begin exchanging encrypted data securely.

For example, a secure communication session over a VPN can be harvested and stored today, then decrypted by an adversary with access to a quantum computer or array of quantum computers at a later date. An adversary with a quantum computer can then break the key establishment part of the handshake and derive the symmetric keys negotiated between the peers. These symmetric keys can then be used to decrypt the encrypted data exchanged between the peers. Any data transmitted today with longer-term confidentiality requirements is already vulnerable [i.2].

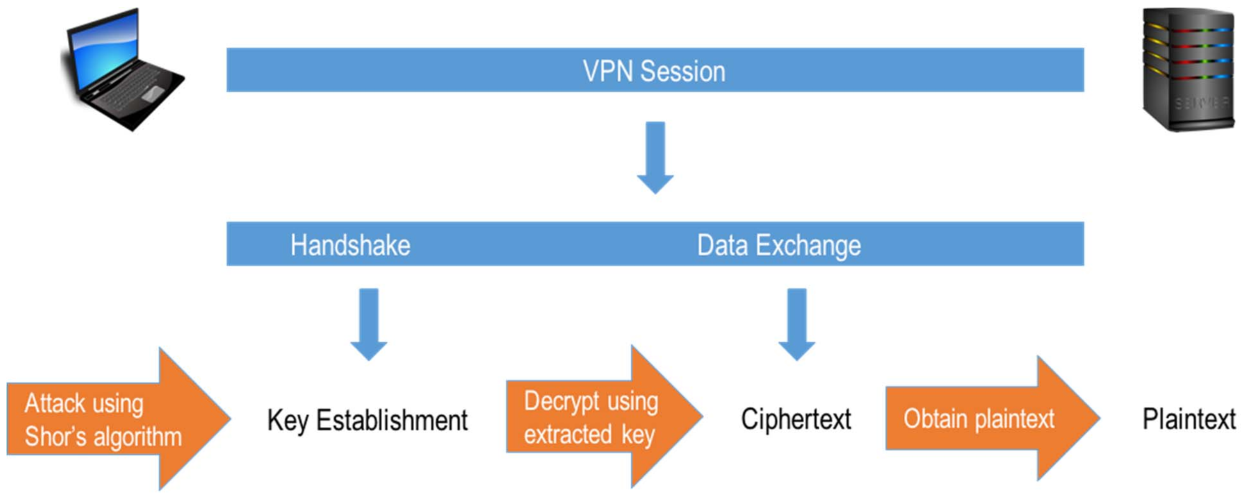

Figure 1: Harvest and Decrypt Attack by a Quantum Adversary

In 2017, quantum safe algorithm candidates were not mature enough to be used on their own. The National Institute of Standards and Technology (NIST) in the United States of America recommends an approach to address this threat today [i.13], by performing a quantum safe key establishment in parallel to a classic key establishment, and then merging the shared secrets before session key generation. In this scenario, if the classic key establishment was performed using a Federal Information Processing Standardization (FIPS) certified module, the entire system would maintain FIPS-certification. For greater assurance, two quantum-resistant schemes can be used in parallel to a classic one. These quantum-resistant schemes need to be based on different hard math problems in case an efficient quantum-based solution is found during the algorithm evaluation process for one of the problems. It should be noted that this style of hybrid cryptography is intended as an interim step, to provide protection of existing systems while the algorithm standardization takes place. Once that standards process is completed, systems are expected to shift to only use quantum safe algorithms.

Quantum safe algorithms differ noticeably from their classical equivalents. Some candidates have significantly larger keys, and in the case of key-encapsulation, larger ciphertext. Some candidates have slower key generation and cryptographic operation times, which impact protocol timing when used in an ephemeral mode for certain applications. All of these properties have an impact on underlying protocols when quantum safe algorithms are used as a replacement for classic equivalents. In the case of hybrid key establishment schemes, this impact is even greater since multiple algorithms are used instead of one.

The purpose of the present document is to clearly describe the range of protocol requirements necessary to add quantum resistance to existing or new implementations of VPNs.

1 Scope

The present document explores protocol requirements necessary to add quantum resistance to VPN technologies, including client, server and architectural considerations. Specifically, requirements around protocols and key establishment are considered, based on the multitude of systems that are at risk and require security updates before quantum computers that can attack commercial cryptography are developed.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI White Paper No. 8, ISBN No. 979-10-92620-03-0: "Quantum Safe Cryptography and Security: An introduction, benefits, enablers and challenges", June 2015.

[i.2] ETSI GR QSC 004 (V1.1.1): "Quantum-Safe Cryptography; Quantum-Safe threat assessment".

[i.3] IETF RFC 4251: "The Secure Shell (SSH) Protocol Architecture".

[i.4] OpenSSH project PROTOCOL file.

NOTE: Available online at http://cvsweb.openbsd.org/cgi-bin/cvsweb/src/usr.bin/ssh/PROTOCOL.

[i.5] IETF RFC 4253: "The Secure Shell (SSH) Transport Layer Protocol".

[i.6] IETF RFC 4252: "The Secure Shell (SSH) Authentication Protocol".

[i.7] IETF RFC 4254: "The Secure Shell (SSH) Connection Protocol".

[i.8] IETF RFC 793: "Transmission Control Protocol".

[i.9] IETF Internet draft: "SSH Agent Protocol draft-miller-ssh-agent-00".

[i.10] IETF RFC 4255: "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints".

[i.11] IETF Internet Draft: "Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2)".

NOTE: Available online at https://tools.ietf.org/id/draft-tjhai-ipsecme-hybrid-qske-ikev2-01.txt.

[i.12] IETF Internet Draft: "Quantum-Safe Hybrid (QSH) Key Exchange for Transport Layer Security (TLS) version 1.3".

NOTE: Available online at https://tools.ietf.org/id/draft-whyte-qsh-tls13-06.txt.

[i.13] NIST Post-Quantum Cryptography FAQs.

NOTE: Available online at https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/faqs.

[i.14] IETF RFC 4301: "Security Architecture for the Internet Protocol".

[i.15] IETF RFC 7296: "The Internet Key Exchange Protocol Version 2 (IKEv2)".

[i.16] IETF Internet Draft: "Postquantum Pre-shared Keys for IKEv2".

NOTE: Available online at https://www.ietf.org/id/draft-ietf-ipsecme-qr-ikev2-02.txt.

[i.17] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[i.18] The Viability of Post-Quantum X.509 Certificates.

NOTE: Available online at https://eprint.iacr.org/2018/063.

[i.19] S. Galbraith, C. Petit, B. Shani and Y. Ti: "On The Security of Supersingular Isogeny Cryptosystems", 2016.

NOTE: Available online at https://eprint.iacr.org/2016/859.pdf.

[i.20] ETSI GR QSC 006 (V1.1.1): "Quantum-Safe Cryptography (QSC); Limits to Quantum Computing applied to symmetric key sizes".

[i.21] IETF Internet Draft: "Quantum-Safe Hybrid (QSH) Ciphersuite for Transport Layer Security (TLS) version 1.2".

NOTE: Available online at https://datatracker.ietf.org/doc/draft-whyte-qsh-tls12/.

[i.22] IETF Internet Draft: "A Transport Layer Security (TLS) Extension for Establishing An Additional Secret".

NOTE: Available online at https://datatracker.ietf.org/doc/draft-schanck-tls-additional-keyshare/.

[i.23] "The Double Ratchet Algorithm".

NOTE: Available online at https://www.signal.org/docs/specifications/doubleratchet/.

[i.24] IEEE 802.1AE-2006™: "Local and Metropolitan Area Networks: Media Access Control (MAC) Security".

[i.25] IEEE 802.1AEbn-2011™: "Local and metropolitan area networks--Media Access Control (MAC) Security Amendment 1: Galois Counter Mode--Advanced Encryption Standard-- 256 (GCM-AES-256) Cipher Suite" (Amendment to IEEE Std 802.1AE-2006).

[i.26] IEEE 802.1AEbw-2013™: "Local and metropolitan area networks-Media Access Control (MAC) Security Amendment 2: Extended Packet Numbering" (Amendment to IEEE Std 802.1AE-2006).

[i.27] IEEE 802.1X-2010™: "Local and metropolitan area networks--Port-Based Network Access Control".

[i.28] IEEE 802.1Xbx-2014™: "Local and metropolitan area networks -- Port-Based Network Access Control Amendment 1: MAC Security Key Agreement Protocol (MKA) Extensions" (Amendment to IEEE Std 802.1X-2010).

[i.29] IEEE 802.1Xck™: "Local and Metropolitan Area Networks - Port-Based Network Access Control Amendment: YANG Data Model". (DRAFT).

NOTE: Available online at https://standards.ieee.org/develop/project/802.1Xck.html.

[i.30] IETF RFC 3394: "Advanced Encryption Standard (AES) Key Wrap Algorithm".

[i.31] IETF RFC 3748: "Extensible Authentication Protocol (EAP)".

[i.32] IETF RFC 5216: "The EAP-TLS Authentication Protocol".

[i.33] IEEE 802.1AR™: "Local and metropolitan area networks-Secure Device Identity".

[i.34] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES	Advanced Encryption Standard
AESKW	Advanced Encryption Standard Key Wrap
AUTH	Authentication
CA	Certificate Authority
CAK	Connectivity Association Key
CERT	Certificates
CKN	Connectivity association Key Name
DA	Destination Address
DH	Diffie-Hellman
EAP	Extensible Authentication Protocol
EAPoL	Extensible Authentication Protocol over LAN
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithms
ETSI	European Telecommunications Standards Institute
FIPS	Federal Information Processing Standardization
HDR	High Data Rate
HTTP	HyperText Transfer Protocol
ICK	Integrity Check Key
ICV	Integrity Check Value
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IKEv1	Internet Key Exchange version 1
IKEv2	Internet Key Exchange version 2
IP	Internet Protocol
IPsec	Internet Protocol security
KDF	Key Derivation Function
KE	Key Exchange
KEK	Key Encryption Key
KEM	Key Encapsulation Mechanism
KN	Key Number
LAN	Local Area Network
LHL	Leftover Hash Lemma
LWE	Learning With Errors
MAC	Message Authentication Code
MACsec	Media Access Control security
MIs	Member Identifiers
MKA	Media access control security Key Agreement
MKPDU	Media access control security Key agreement Protocol Data Unit
MSK	Master Session Key
MTU	Maximum Transmission Unit
NATs	Network Address Translators
NIST	National Institute of Standards and Technology
PKI	Public Key Infrastructure
PPK	Post-quantum Pre-shared Key
PRF	PseudoRandom Function
PSK	Pre-Shared Key
QSC	Quantum-Safe Cryptography
QSH	Quantum Safe Hybrid
QS_SA	Quasi Steady - State Approximation
QS_KE	Quantum-Safe - Key Exchange
RFC	Request For Comments
RSA	Rivest Shamir Adleman
RTT	Round Trip Time
SA	Security Association
SAK	Security Association Key

SC	Secure Channel
SIDH	Supersingular Isogeny Diffie–Hellman
SK	Secret Key
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSHFP	Secure Shell key Fingerprint
TCP	Transmission Control Protocol
TLP	Transport Layer Protocol
TLS	Transport Layer Security
US	United States (of America)
VPN	Virtual Private Network

4 General Virtual Private Network (VPN) requirements

4.1 Background

The primary purpose of a VPN is to provide a secure connection between two end points. While the specific technologies used to accomplish this can vary widely, the general goals are similar. A VPN then is the technology used to construct a private network over public channels. Large organizations typically consist of different locations that are geographically widespread. Each location can have its own Local Area Network (LAN) within which many servers and client computers interconnect. Nonetheless, it would be necessary to connect LANs of different locations, and some remote entities, to provide a single network service for the entire organization. Different locations can be connected by separate, dedicated, and well-protected communication lines. However, as the number of sites increases, and they become more physically separated, such a solution becomes cost prohibitive. Supporting access to this private network by remote entities, such as traveling employees, can become infeasible since the number is extremely large and remote entities can move. VPN allows the use of public networks to connect all these locations and entities as a single private network.

Often, sensitive data is transmitted over a private network among internal servers, or between an internal server and an internal client. Therefore, the security of data in transit over a VPN is critical. Since current VPN technologies utilize public key cryptography extensively for its security, they are vulnerable against quantum attacks.

VPN architectures are often categorized into two connection types: Site-to-site VPN and Remote Access VPN. Because of the different characteristics of these two types of VPN, the requirements differ.

Site-to-Site VPN

The site-to-site VPN establishes a secure tunnel between the local private networks of two physically separated locations over public networks, such that the network behaves as if it is a single private network. An outsider of the VPN can observe the existence of data traffic between the two sites, but cannot see who is connecting with whom, or read the information in the data traffic. VPN was originally developed for this purpose. In addition, the VPN provides authentication between the two parties.

The site-to-site VPN connects two sites semi-permanently, therefore, tunnel establishment might not be performed frequently. Since the tunnel is spanned between only two gateways, flexibility to meet the needs of different deployments is often preferred over the cost of complexity. Also, it is usually not overly difficult to set up a shared secret between the two gateways or static self-signed certificates.

Remote Access VPN

The remote access VPN allows a personal device to connect into the organization's private network over the public network such that the computing resources on this private network become available to the remote device. An outsider can see the data traffic between the remote device and the gateway but cannot identify which computing resource in the private network the remote device is connecting to or read or modify the information in the data traffic.

In contrast with site-to-site VPN, tunnel establishments occur very frequently in the remote access VPN. Also, client (device) authentication is critical for remote access VPN. This is, in part, because when the number of client devices is large, there is key management required to securely share secrets between the devices and the gateway.

Underlying Security Protocols and Quantum Vulnerabilities

VPN achieves cryptographic security by the underlying security protocols. These include Internet Protocol Security (IPSec) and Internet Key Exchange (IKE); Transport Layer Security (TLS); Media Access Control Security (MACsec); Secure Shell (SSH), and others.

Some VPN protocols are used to establish security between network entities, i.e. at the network layer level, and some between applications. All protocols accomplish data confidentiality and authentication using symmetric algorithms. Most protocols accomplish entity authentication and key establishment using public key cryptography, while some rely on pre-shared secrets. While public key based key establishment algorithms are typically negotiated between peers and make phased system upgrades possible, the authentication algorithms are not typically negotiated with as much flexibility since public key certificates contain only one public key type which makes system upgrades difficult. However, some protocols have introduced more flexibility here such as TLS 1.3 [i.34].

Generally speaking the needs of each protocol listed above are relatively similar at a base level - confidentiality and authentication - but how those are implemented may be very different depending on the use case. For example, when TLS is used to establish a VPN between multiple corporate users and the head office, it may require client authentication while TLS used between a web client and a public web server may only require server authentication. Some of these specific needs are addressed in the relevant clauses later.

Public Key Infrastructure (PKI)

Presenting a public key with a digital signature only proves that the sender owns the corresponding private key and is not sufficient to establish entity authentication. It is necessary to construct a system that can prove or assure that the presented public key belongs to a legitimate entity. PKI is developed to provide such a mechanism of public key-based authentication. Digital signature algorithm is the foundation of PKI used to achieve cryptographic security. In PKI, a trusted third party, a Certificate Authority (CA), vets the identity of an entity and its public key. It then composes a digital certificate that contains the identity and the public key of the entity, and digitally signs the certificate. An entity can then present its digital certificate and use the corresponding private key to generate a signature on a random challenge to prove its identity. The questioning party can then verify the digital signature on the challenge to confirm the entity has possession of the private key, associated with the certificate. They can also validate the certificate by verifying the CA's signature on the certificate. PKI is essential for establishing authentication in many protocols. While it is possible to pre-install all certificates on entities in a closed environment, a PKI is still necessary for effective certificate revocation checking.

4.2 Requirements for hybrid use cases

Security, authenticity and forward secrecy against classical computers are inherent from the classical handshake mechanism. In this transition period as the new quantum-safe algorithms are standardized, there is a desire to keep the properties offered by existing classical handshakes but adding protection from quantum computers. As a result, the use of a hybrid scheme then provides quantum security and quantum forward secrecy while maintaining the classical handshake properties.

Broadly speaking, the two main areas of concern from a quantum computer are in regard to confidentiality and authentication. Confidentiality is of a higher priority risk due to the threat of "harvest and decrypt" and so requires changes early. Authentication is a lower priority concern today but may involve a complex migration path. This priority is also highlighted by the fact that confidentiality risk today is from a passive attacker, utilizing a quantum computer in the future, while the authentication threat is from an active attacker with a quantum computer.

Hybrid schemes generally work by combining keys established via a classical cryptographic protocol, using RSA or ECC, with keys established via one or more quantum-safe protocols, e.g. Lattice-based, Isogeny-based or others. From a security perspective, this is a "best of both worlds approach". The reason for this is that the hybrid solution continues to provide protection from classical attacks today, as existing schemes are mathematically hard for existing computers, as well as adding forward protections against future quantum attacks through the use of quantum-safe schemes. By combining the output of these schemes to produce the shared key, potentially through the use of a Key Derivation Function (KDF), then the security of either is neither reduced nor diluted. In addition, the use of multiple quantum-safe protocols in a hybrid scheme, such as Lattice-based with an Isogeny-based scheme, provide further risk mitigation as the system then relies upon the security of multiple hard math problems. If a classical or quantum algorithm is developed to break the security of Learning with Errors (LWE), for example, then the system is still protected by the Isogeny scheme. This of course needs to be balanced with the potential complications introduced with additional calculations and additional bandwidth usages.

Another topic of concern is the use of PKI for the remote access VPN as it can be complex when used for a large number of members in an organization. It can have layers of subordinate CAs, and the PKI system can serve for multiple applications beyond VPN. Traditional PKIs can only support a single algorithm in digital certificates which makes it very rigid. A new mechanism is needed that supports both traditional RSA or ECC signatures, as well as quantum-safe signatures, and in a way that is backwards compatible with existing clients. Such a mechanism would allow migrating a PKI system to be quantum safe.

The present document focuses on the confidentiality aspects of VPN technologies.

While each specific protocol below has specific needs related to the use of hybrid schemes, the following common requirements are identified as follows:

- Need for quantum-safe cryptography now:
  - Reason: As a result of continued academic and industrial progress, cryptographically relevant quantum computers are coming. Some applications, such as healthcare, government and automotive need to prepare today for that threat given the long lifetime of their information and length of time to upgrade their systems. The focus is on public key cryptography as symmetric key cryptography appears to be safe as long as keys are long enough [i.20].
- Ensure backwards compatibility:
  - Reason: One of the main reasons to employ a hybrid scheme today is to introduce protections to quantum computers in existing systems as a risk mitigation technique. It is important for a hybrid scheme to provide backwards compatibility so that quantum-safe algorithms can be introduced to existing systems while still allowing interoperability with systems that only support classical algorithms. This ensures the maximum ease of adoption of new quantum-safe algorithms.
- Ensure FIPS compliance:
  - Reason: Besides Europe's Common Conformity framework for ICT security products, NIST's FIPS framework is widely used as a purchasing requirement by European and North American customers so preparing for FIPS compliance is wise. However, the algorithms that are believed to be quantum-safe are not FIPS compliant yet. Still, the end requirement is that the overall hybrid quantum safe design be FIPS compliant, for the relevant classical cryptographic portions.
- Provide cryptographic agility and do not create assumptions about the properties of algorithms within the protocols:
  - Reasons: If vulnerability is discovered in one of the algorithms, or if an algorithm with different properties is needed in the future, then it can be easily replaced. If algorithms with different properties become necessary in future, this framework can be used unchanged to facilitate migration to those algorithms.
- Limit the amount of exchanged data:
  - Reason: It is important for a hybrid scheme to limit the amount of data exchanged, particularly when multiple public keys are involved. This data limitation is often imposed by the specific protocol related to record size limits, fragmentation rules or other requirements. However, this needs to be balanced with the computational requirements as reducing the amount of data sent by performing expensive computations would not be appropriate.

In addition, the following design and user experience recommendations are identified for implementers:

- Use at least 2 algorithms; security key handshake to be negotiated over a combination of at least one classical handshake and one or more quantum-safe handshakes.
- Minimize user experience impact:
  - Reason: It is expected that the computational overhead of any calculations do not cause a noticeable delay on the end user experience. This will depend on the application however, such as TLS being used for site-to-site VPN traffic versus being used for a remote access VPN. In addition, the amount of data transferred is also considered in how it can impact the user experience, for example in the context of time to load data.
- Localize and limit changes to protocols:
  - Reason: Limiting changes in the exchanged messages and state machine is important, so that they may be easily implemented, reviewed and verified.
- Focus on confidentiality:
  - Reason: A passive attacker can eavesdrop encrypted communications today and decrypt it once a quantum computer is available in the future. This is a very serious attack with no solution. An attacker can only perform active attacks such as impersonation of the communicating peers once a quantum computer is available, sometime in the future. Thus, the design focuses on quantum-resistant confidentiality due to the urgency of this problem.
- Have an efficient negotiation of hybrid algorithm -- comprising multiple quantum-safe algorithms:
  - Reason: In some protocols, the particular combination of algorithms to use is specified by a cipher suite or other name. In those protocols, the usage of a hybrid approach in which each hybrid combination of several classical and quantum-safe algorithms leads to a different identifier can lead to a large number of combinations. Efficient algorithm negotiation is then key. The efficiency is considered in the context of number of round trips to complete the negotiation and the amount of data transferred and processed to facilitate this.

4.3 Direct drop-in requirements

Rather than using a hybrid solution, new quantum-safe key exchange algorithms could be added to the VPN protocol. They can be used directly in the particular key exchange or setup process inherent to the VPN and potentially negotiated in a similar way. The main concerns in this context are whether the quantum-safe algorithms will fit within the constraints of the protocol and, perhaps more importantly, is that most quantum-safe key exchange algorithms are relatively new and require further study.

While each specific protocol below has specific needs related to the use of quantum-safe schemes, the common requirements as well as design recommendations identified above for hybrid schemes are also valid for direct drop-in quantum-safe schemes.

5 Internet Key Exchange (IKE) based requirements

5.1 Background

IPsec [i.14] is the one of the most popular protocols used by VPNs. Peer authentication and key establishment is performed by IKE protocol [i.15], which is used with IPsec. IKE performs key establishment and creates an encrypted channel before the peers are authenticated to protect peer identities from eavesdroppers.

IKE comes in two flavours, the older and now less used IKE version 1 (IKEv1) and the newer and more widely deployed IKE version 2 (IKEv2). The security of IKEv2 is based on the Diffie-Hellman (DH) exchange, which can be broken by a quantum computer; once the IKE security association is established, the authentication exchange occurs. Currently all authentication methods (i.e. RSA and Elliptic Curve Digital Signature Algorithms (ECDSA)) used for the authentication exchange, except symmetric-key based, are broken by a quantum computer.

IKEv1 does not solely rely on the DH exchange. In fact, when using IKE with a pre-shared-key for authentication, the contents of the pre-shared-key are mixed into creating the shared-secret, hence IKEv1 with a pre-shared-key for authentication can be considered quantum safe [i.16].

There are two properties of this protocol that make adding a quantum-safe key establishment scheme difficult. To begin with, the messages of the first exchange that carries key establishment material will be fragmented at the IKE layer and will be fragmented at the Internet Protocol (IP) layer if they are too long. Fragmented IP packets are typically filtered out by Firewalls, Proxies, Load Balancers and Network Address Translators (NATs), which understandably breaks IKE. This puts a limit on the size of the public keys that can be sent. Even not all classic DH schemes are suitable for IKE use, due to the size of their public keys. The larger public-key sizes for quantum-safe algorithms may not work for IKE, either as part of a hybrid scheme or as a direct drop-in, as they will make the initial messages too long. A mechanism to allow fragmentation of these larger packets will be needed.

Secondly, IKE essentially "guesses" the DH parameters supported by the peer and sends the public key material in the first message. If the peer does not support the selected parameters, it notifies the peer and new parameters are sent in the next message exchange, resulting in a wasted pair of messages. As many quantum-safe algorithms are larger in size compared to current DH parameters, wasting a message exchange using a quantum-resistant algorithm that is not supported by the peer would be wasteful in some circumstances.

To support a phased upgrade of peers, a more efficient solution is needed when an upgraded client is attempting to connect to a legacy server.

When a VPN session is first established, two security associations (SAs) are established. The IKE SA is established first in the IKE_SA_INIT exchange, which is used to negotiate all further SAs. Next, the first child SA is established in the AUT_AUTH exchange. The child SAs handle the secure transport of client and server data. Further child SAs can be created, or existing SAs can be rekeyed in optional CREATE_CHILD_SA exchanges.

If during IKE SA establishment or subsequent child establishment or rekeying the initiator's guess for key exchange data is not what the responder prefers, the responder will reply with the preferred key exchange algorithm and the initiator will resend the IKE_SA_INIT or CREATE_CHILD_SA message with the desired key exchange data.

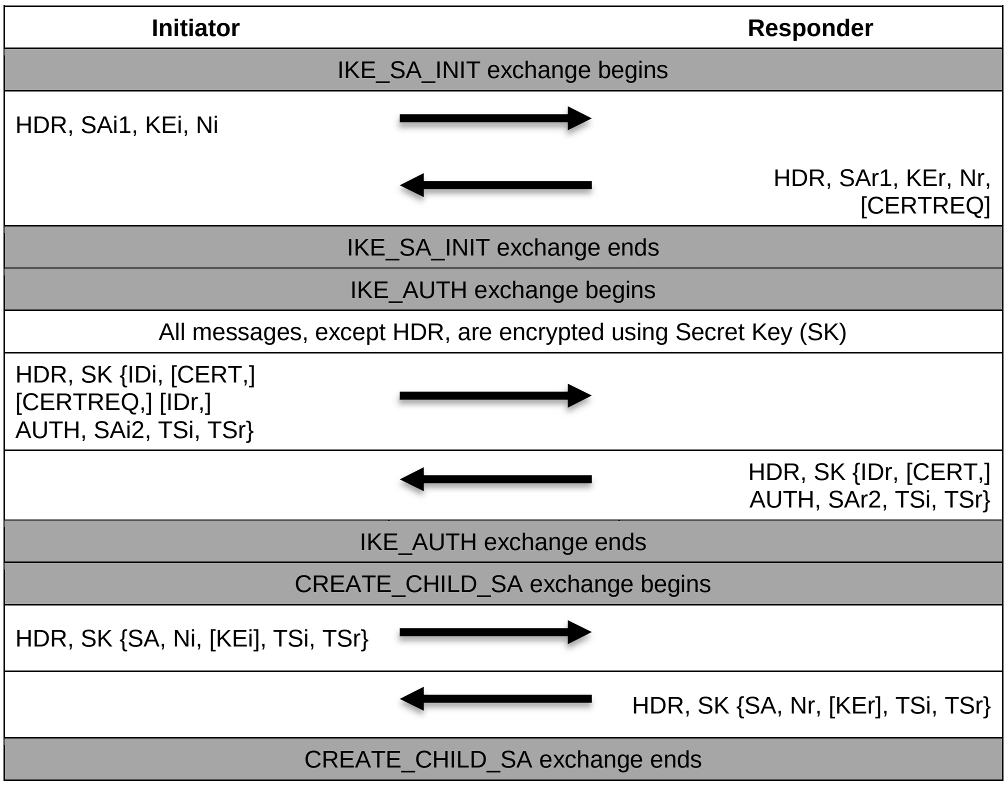
Initiator
Responder
IKE_SA_INIT exchange begins
HDR, SAi1, KEi, Ni
HDR, SAr1, KEr, Nr, [CERTREQ]
IKE_SA_INIT exchange ends
IKE_AUTH exchange begins
All messages, except HDR, are encrypted using Secret Key (SK)
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr}
HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}
IKE_AUTH exchange ends
CREATE_CHILD_SA exchange begins
HDR, SK {SA, Ni, [KEi], TSi, TSr}
HDR, SK {SA, Nr, [KEr], TSi, TSr}
CREATE_CHILD_SA exchange ends

Figure 2: IKEv2 Handshake

5.2 Hybrid requirements and solutions

5.2.1 Requirements

In addition to the requirements discussed in clause 4.2, the following requirement is identified for the design of a hybrid quantum-safe solution.

- Support fragmentation support of key shares:
  - Reason: Some quantum-safe algorithms could be relatively bulky and they might require fragmentation. As a result, it is necessary to utilize the adaptation and adoption of an existing fragmentation method or the design of a new method that allows for the fragmentation of the key shares.

The use of a new transform type is an approach that could be considered for hybrid algorithms but it may cause problems with existing IKE implementations. Further discussion of hybrid requirements for IKE can be found in [i.11].

5.2.2 Solutions

5.2.2.1 Introduction

Clause 5.2.2 documents multiple approaches to include hybrid quantum-safe key exchange algorithms within IKE. Each of these has different trade-offs in terms of setup time, fragmentation and anonymity and how they protect against downgrade and denial-of-service attacks.

5.2.2.2 Establish an IKE connection as usual

The first approach is to establish an IKE connection as usual, and then rekey the SAs in the CREATE_CHILD_SA exchange, using quantum-safe keys. Support for this approach is negotiated in the IKE_SA_INIT exchange. The QS_ALGOS payload contains the proposed and selected quantum-safe algorithms, and the QS_KE payloads contain the quantum-safe key exchange data. The shared secret resulting from the quantum-safe key exchange is concatenated with the shared secret from the classical key exchange before deriving the keying material.

This approach does not preserve anonymity. A quantum attacker would be able to determine the identities of the endpoints since the authentication phase is not protected by a quantum-safe key exchange.

This approach avoids IP fragmentation issues because the large quantum-safe key exchange data is sent after the IKE_SA_INIT exchange, so the endpoints are able to take advantage of IKEv2's fragmentation support.

A variation of this approach negotiates the quantum-safe algorithms during the IKE_SA_INIT phase rather than in the QS_ALGOS payload of the CREATE_CHILD_SA phase. In this way, the initiator does not need to guess which algorithm the responder will choose, avoiding the initiator to guess wrong, which it would have performed a potentially expensive key exchange data generation unnecessarily. The actual quantum-safe key exchange still does not occur until the rekeying phase though.

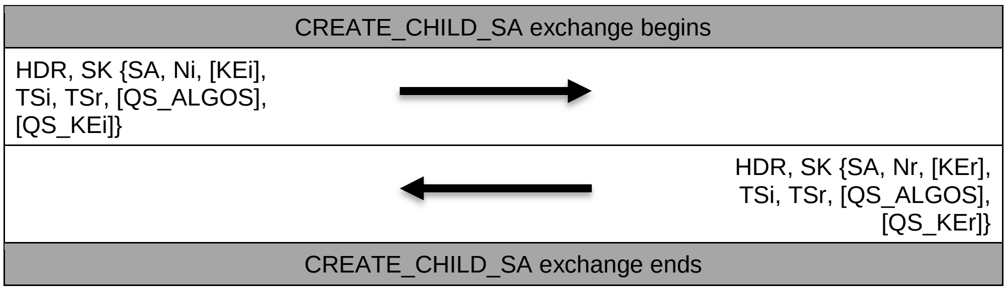
CREATE_CHILD_SA exchange begins
HDR, SK {SA, Ni, [KEi], TSi, TSr, [QS_ALGOS], [QS_KEi]}
HDR, SK {SA, Nr, [KEr], TSi, TSr, [QS_ALGOS], [QS_KEr]}
CREATE_CHILD_SA exchange ends

Figure 3: Hybrid IKEv2 Rekey

5.2.2.3 Add a new IKE_QS_KE phase between the IKE_SA_INIT and IKE_AUTH phases

The second approach is to add a new IKE_QS_KE phase between the IKE_SA_INIT and IKE_AUTH phases. Support for this approach is negotiated in the IKE_SA_INIT exchange. The quantum-safe key exchange algorithm negotiation occurs in the QS_SA payloads and the quantum-safe key exchange is transferred in the QS_KE payloads. At the end of this exchange, the IKE SA's keying material is regenerated using a combination of the previous classical shared secret and the new quantum-safe shared secret, thus any further exchanges will be protected by the quantum-safe key exchange in this phase.

This approach preserves anonymity against a quantum attacker because the authentication occurs after the quantum-safe shared secret is mixed into the keying material. IP fragmentation issues are also avoided because the quantum-safe key exchange takes place after the IKE_SA_INIT phase so IKEv2 fragmentation can be used.

A variation of this approach negotiates the quantum-safe algorithms during the IKE_SA_INIT message rather than in the QS_SA payload. This allows the initiator to be sure of which quantum-safe algorithm the responder will select and avoids a potential unnecessary quantum-safe key generation.

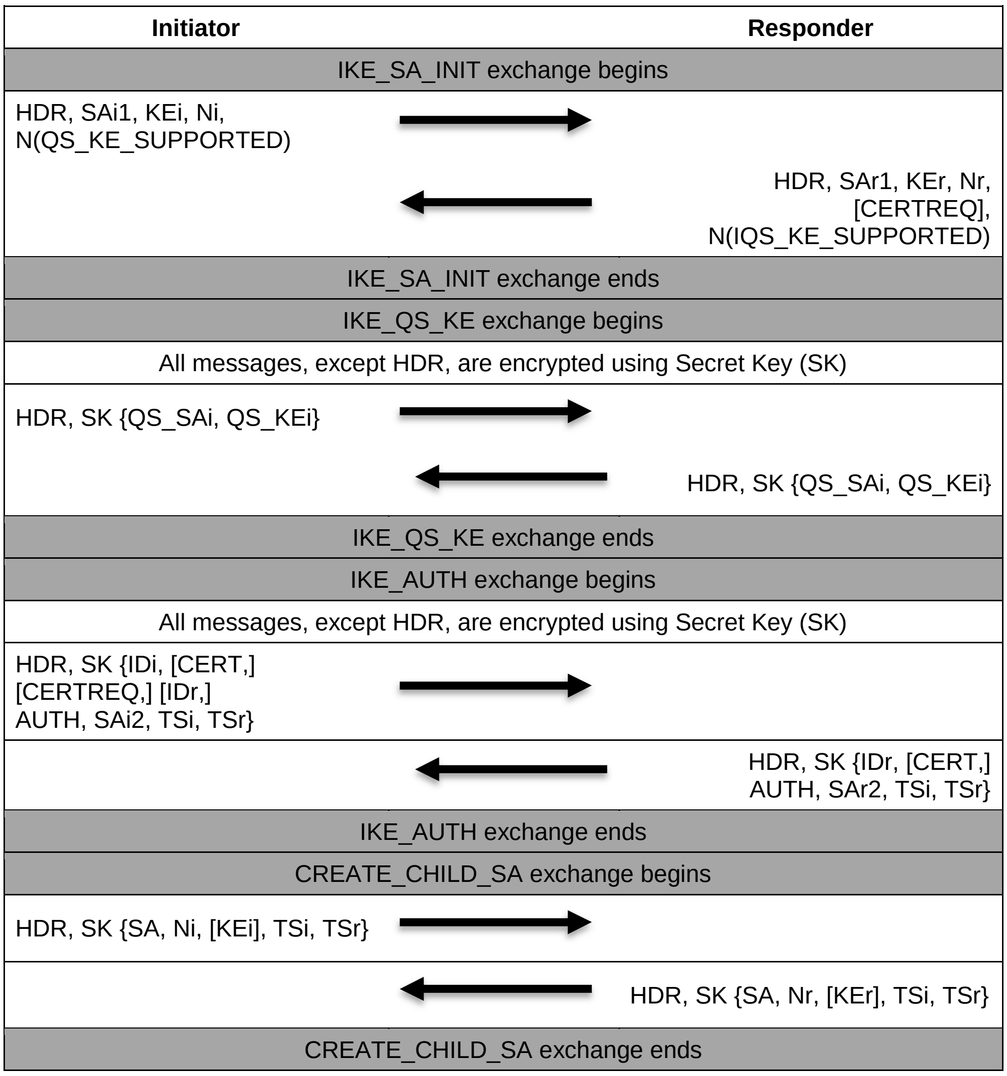
Initiator
Responder
IKE_SA_INIT exchange begins
HDR, SAi1, KEi, Ni, N(QS_KE_SUPPORTED)
HDR, SAr1, KEr, Nr, [CERTREQ], N(IQS_KE_SUPPORTED)
IKE_SA_INIT exchange ends
IKE_QS_KE exchange begins
All messages, except HDR, are encrypted using Secret Key (SK)
HDR, SK {QS_SAi, QS_KEi}
HDR, SK {QS_SAi, QS_KEi}
IKE_QS_KE exchange ends
IKE_AUTH exchange begins
All messages, except HDR, are encrypted using Secret Key (SK)
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr}
HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}
IKE_AUTH exchange ends
CREATE_CHILD_SA exchange begins
HDR, SK {SA, Ni, [KEi], TSi, TSr}
HDR, SK {SA, Nr, [KEr], TSi, TSr}
CREATE_CHILD_SA exchange ends

Figure 4: Hybrid IKEv2 IKE_QS_KE Phase

5.2.2.4 Utilizing the IKE_SA_INIT message

This approach is to do the quantum safe key exchange in the IKE_SA_INIT message. The existing SA and Key Exchange (KE) payloads can be modified to include combined classical plus quantum-safe key exchange algorithms, or new payloads or notifications can be added to transport the quantum-safe proposals and key exchange data. The classical and quantum-safe shared secrets are combined before generating the SA keying material. This hybrid key exchange is negotiated similarly to how the DH key exchange is currently negotiated in IKEv2.

This approach preserves anonymity and prevents downgrade attacks, but it does not handle IP fragmentation. Large IKE_SA_INIT messages can be broken into IP fragments and these fragments can be dropped in transit. Additional changes to the IKEv2 protocol would be needed to handle IP fragmentation issues in the IKE_SA_INIT exchange.

5.2.2.5 Framework of hybrid quantum-safe key exchange

This approach is to make IKEv2 [i.15] quantum-safe via duplicating the initial exchange in IETF RFC 7296 [i.15]. More details on the specific proposal can be found in [i.11].

The key part of this approach is a focus on minimizing communications overhead by limiting the data portion to key shares. To do this, the IKE_SA_INIT exchange is updated to include two message exchange pairs. The first pair is focused on the classical algorithm negotiation and along with which quantum-safe algorithms each side support, and any other related policies. That then allows the second pair to include both the DH public value and the quantum-safe key shares.

This hybrid key exchange can occur in the IKE_SA_INIT message or alternatively in the CREATE_CHILD_SA message pair. This allows for a flexible structure.

Additional details included in [i.11] are:

a) negotiation of fragmentation in the first round and mechanism for performing fragmentation;

b) logic for dynamic hybrid group negotiation; and

c) methods to prevent downgrade attacks.

5.3 Direct drop-in requirements

Rather than using a hybrid solution, new quantum-safe key exchange algorithms could be added to the IKEv2 protocol. They can be used directly in the SA and key exchange payloads instead of the existing classical algorithms by defining new key exchange identifiers and are negotiated the same way that existing DH algorithms are negotiated. The main challenge here is that most quantum-safe key exchange algorithms are relatively new and require further analysis.

Clause 5.2.1 lists a number of requirements applying to a hybrid solution. Most of those points also apply when only interested in a drop-in solution.

In particular, IKEv2 messages using these new algorithm IDs will be subject to IP fragmentation issues since the quantum-safe key exchange data is typically larger than the MTU path. If the IP fragmentation causes IP fragments to be dropped, then the VPN endpoints will have to negotiate a way to avoid this. The endpoints could choose to perform TCP encapsulation of the IKEv2 SA negotiation, although this approach can introduce additional issues such as increased latency or TCP being disallowed on the network. Alternatively, a method to fragment IKE_SA_INIT messages could be developed which avoids IP fragmentation (for instance, see method in clause 5.2.2.4 or directly in [i.11]).

Similar to IKEv2, IKEv1 can use new algorithm IDs to add new key agreement algorithms, however these will also be subject to IP fragmentation. IKEv1 can also use its pre-shared keys feature to create non-hybrid quantum-safe keys.

In addition to the requirements in clause 4.2, the following requirements are identified for the design of a direct drop-in quantum-safe scheme:

- Concentrate changes on IKEv2, instead of IKEv1, given the prevalence of it.

5.4 Quantum-safe pre-shared keys for IKEv2

An alternative to using a quantum-safe key exchange would be to use pre-shared keys. In this approach, it is assumed that both sides share a common secret string; the standard IKE protocol would be performed (using a DH or an ECDH key exchange), and would stir in this long secret into the keys that will protect traffic. If this value is sufficiently long to make Grover's algorithm infeasible, then these keys will be quantum safe. This is a viable option in many IKE deployments as IKE is often used in VPN scenarios, where the two sides are relatively static. This approach has the advantage that it does not rely on the conjectured hardness of an asymmetric quantum-safe algorithm; it is quantum secure as long as the KDF and the traffic encryption algorithms (which are symmetric) are quantum secure (which is an assumption anyways).

As mentioned previously, when the previous version of IKE (IKEv1) is configured to do authentication using pre-shared keys, this happens automatically; only a sufficiently long pre-shared key is needed. However, the newer version (IKEv2) is preferred, as it is simpler, more reliable, has better extension support, and is easier to diagnose on a configuration mismatch.

IKEv2 does not implement this when it uses pre-shared keys for authentication. One proposed extension to implement this in IKEv2 is in [i.16]. In this protocol extension, both the initiator and the responder share a secret value (a Postquantum Pre-shared Key or PPK) along with a PPK identifier; this PPK is independent of the authentication methods that IKEv2 uses.

The negotiation is outlined in Figure 5; during the initial exchange, both sides exchange the N(USE_PPK) notification to indicate support for this extension. Then, in the IKE_AUTH message, the initiator constructs the IKE SA as usual, the initiator uses its copy of the PPK to both update its internal SK_d, SK_pi, SK_pr values, and generates the auth payload based on these updated value. Then, it sends that (along with the identifier of the PPK it used). If the responder recognizes the PPK identifier, it can use its copy of the PPK to validate the auth payload; it then updates its copy of the SK_d. The SK_d value that both sides have updated is then used to generate the keying data for any later SAs (including the IPsec SAs), hence any child SAs are quantum-safe.

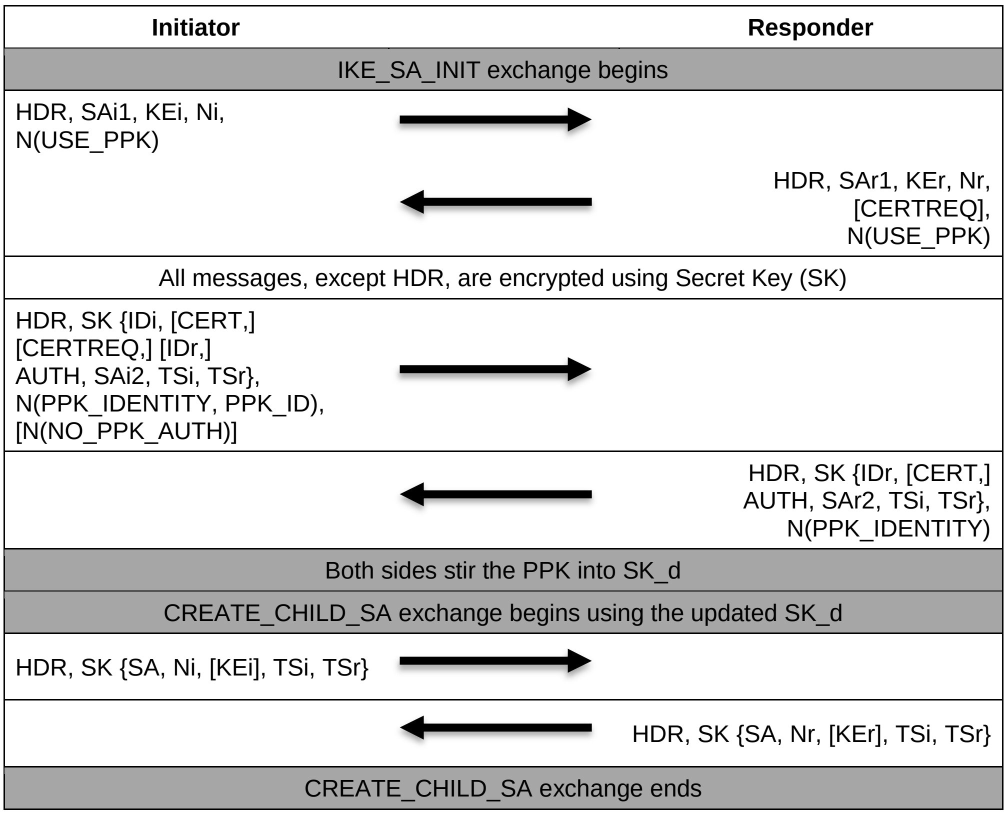
Initiator
Responder
IKE_SA_INIT exchange begins
HDR, SAi1, KEi, Ni, N(USE_PPK)
HDR, SAr1, KEr, Nr, [CERTREQ], N(USE_PPK)
All messages, except HDR, are encrypted using Secret Key (SK)
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr}, N(PPK_IDENTITY, PPK_ID), [N(NO_PPK_AUTH)]
HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}, N(PPK_IDENTITY)
Both sides stir the PPK into SK_d
CREATE_CHILD_SA exchange begins using the updated SK_d
HDR, SK {SA, Ni, [KEi], TSi, TSr}
HDR, SK {SA, Nr, [KEr], TSi, TSr}
CREATE_CHILD_SA exchange ends

Figure 5: IKEv2 Handshake using PPK

This protocol has the advantage that it deals with PPK mismatches cleanly, allowing the two sides to either fall back to a non-quantum-safe mode, or refuse the exchange (depending on policy). It has the further advantage that it has minimal expansion of the IKE packets, and hence it does not need to address fragmentation. Note that, in this approach, the initial IKE SA is not quantum secure; however any later SAs (including the IPsec SAs) are. One effect is that this approach does not preserve anonymity against quantum-capable adversaries (as the IKE SA that encrypts the identities is not quantum safe).

This general approach to achieve quantum resistance by using pre-shared keys has some disadvantages. For one, it requires someone to configure the PPK on both sides (and this configuration will presumably be done in a quantum safe manner). In addition, it does not implement perfect forward secrecy against someone who can break the DH exchange inherent in IKE (which a Quantum Computer is able to). The impact of having no forward secrecy is that one attack at a single point in time is sufficient to compromise all communication. This includes scenarios where the shared secret is compromised after distribution or use. There it is important to protect pre-shared secrets from generation until destruction. This can be mitigated by frequent distribution of shared secrets or other mitigation techniques such as hashing the shared secret each time it is used and/or mixing in DH exchanges with the shared secret [i.23].

6 Transport Layer Security (TLS) based requirements

6.1 Background

TLS [i.17] is another popular protocol used in VPNs. It also consists of the handshake part and the data exchange part executed between client and server. There are several TLS versions, TLS 1.3 [i.34] being the most recent one approved in April 2018. The operation of the TLS handshake depends on the TLS version. In TLS 1.2 [i.17] the algorithms used for key establishment are negotiated before the key establishment key material is exchanged (see Figure 6). In the most recent TLS 1.3 [i.34] the client directly sends its key shares to the server in its first message so that the server can pick up one and perform key agreement in a single round trip (see Figure 7) and it can also start sending data much earlier. In some cases where the client and the server do not share a common parameter set, the two parties may need to negotiate the parameters prior to the handshake adding an additional round trip so that the performance is equivalent to the one of TLS 1.2 [i.17]. This is depicted in Figure 8.

The messages of the handshake and the payload are transported in the so called record layer. A potential source of problem here is the record size. Everything in TLS, including the handshake and data messages, is sent using records that cannot exceed 16 kilo octets. This is specified in clause 6.2.1 in TLS 1.2 [i.17], and clause 5.1 in TLS1.3 [i.34]. A typical handshake message is under 16 kilo octets, but with additional key material the messages will likely exceed 16 kilo octets. In the TLS 1.2 specification [i.17], improper handshake message fragmentation is listed as a common implementation pitfall.

Timing is also impacted. Additional key establishment operations in hybrid mode will increase the time required to complete the handshake. Depending on the additional time required, this can impact the dependant application layer.

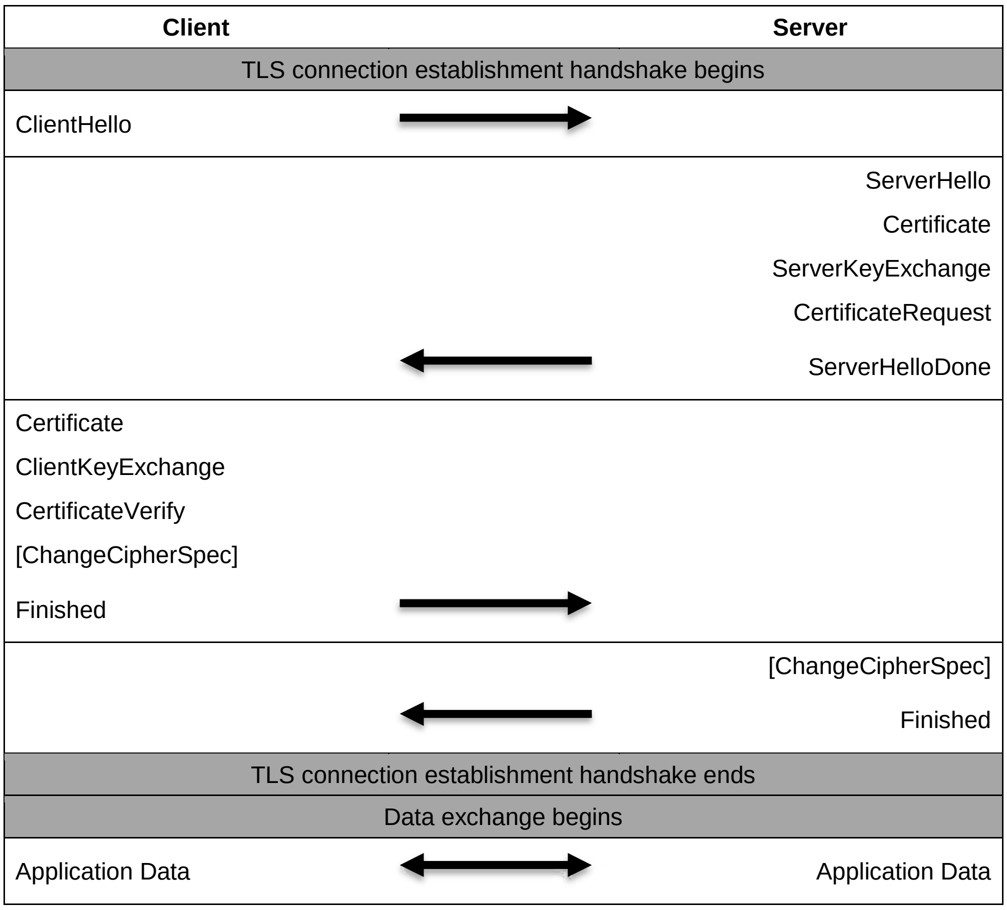

Figure 6: TLS 1.2 [i.17] Handshake

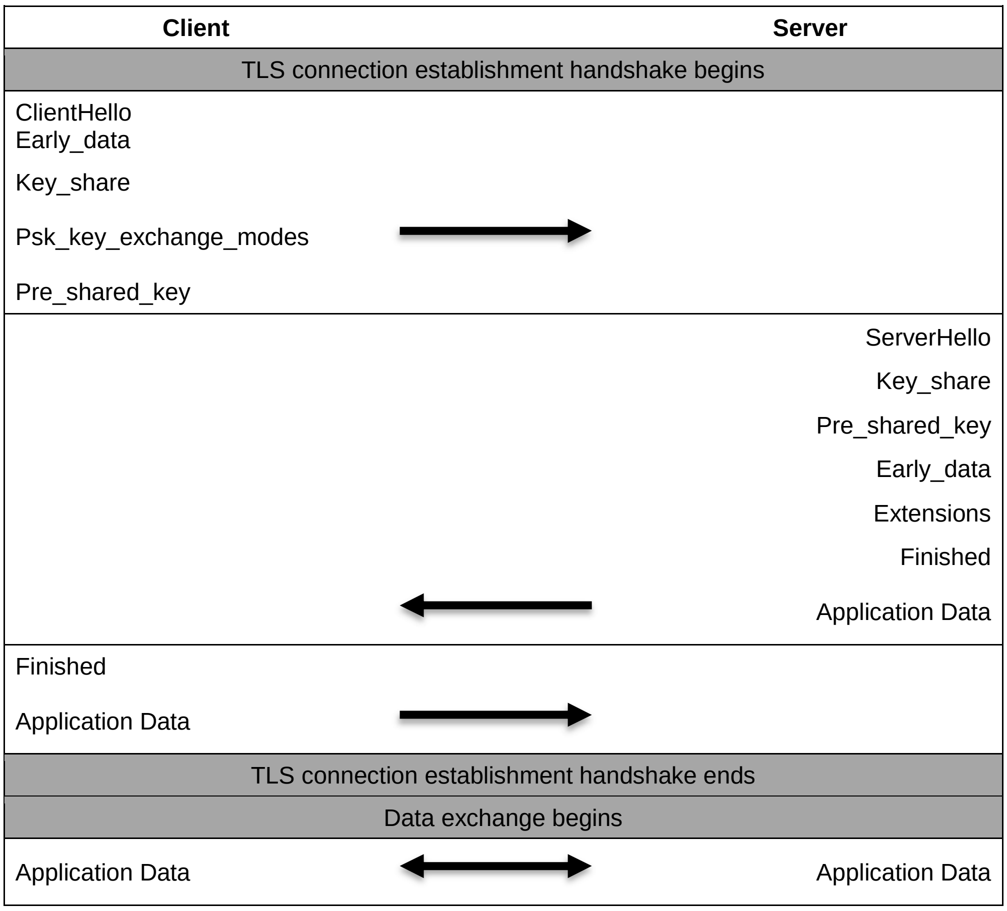

Figure 7: A Full 0-RTT Handshake in TLS 1.3 [i.34]

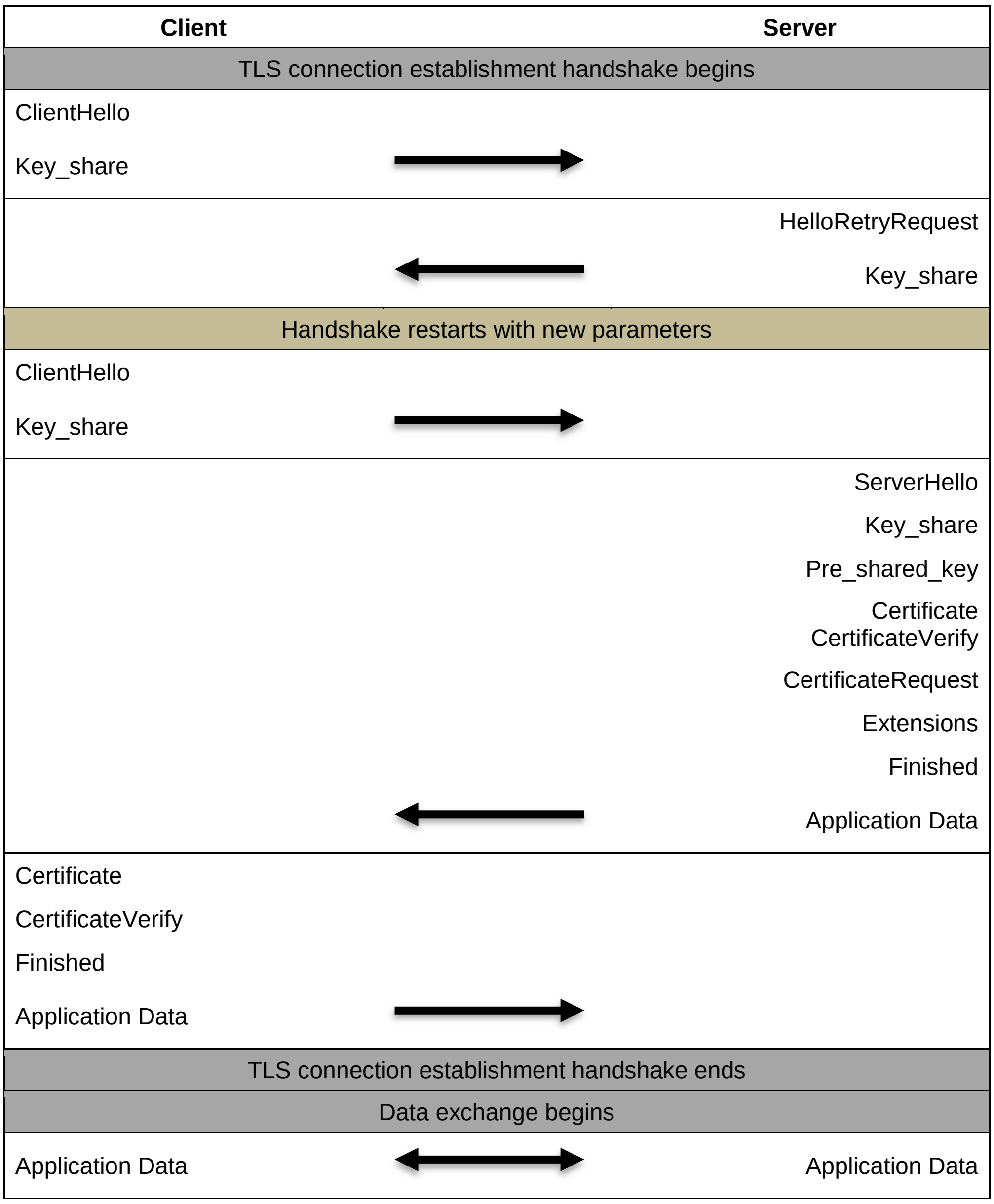

Figure 8: A Full Handshake With Mismatched Parameters in TLS 1.3 [i.34]

6.2 Hybrid requirements and solutions

6.2.1 Requirements

In addition to the requirements listed in clause 4.2, the following requirements are identified for a hybrid solution, mainly as outlined in [i.12]:

- Limit changes to TLS 1.3:
  - Comply with TLS 1.3 0-RTT handshake.
  - Limit changes to enable quantum-safe handshake.

- Define an efficient identification of hybrid algorithms:
    - Reason: The usage of a hybrid approach in which each hybrid combination of several classical and quantum-safe algorithms leads to a different group identifier can mean an exponential growth of identifiers and lack of interoperability. Using complex hybrid schemes can also make the TLS state machine complex.

6.2.2 Solutions

There have been several proposals at the Internet Engineering Task Force (IETF) to realize hybrid quantum-safe TLS solutions. These proposals include extensions to TLS 1.2 [i.21] and TLS 1.3 [i.12] and [i.22].

The present clause documents hybrid and modular approaches to including new quantum-safe key exchange algorithms within TLS 1.3 [i.12], while maintaining the assurance that comes from the use of already established cipher suites. It allows the TLS premaster secret to be agreed upon using both an established classical DH key exchange and a quantum-safe key exchange mechanism.

The general design is to reuse the existing handshake design for DH and ECDH groups, treating the quantum-safe key exchanges as additional ECDH groups as much as possible. In addition, the design provides for the ability to negotiate several key exchanges at the same time (which could include both a classical ECDH group, and a quantum-safe key exchange) and then combine the outputs of the key exchanges through a single KDF. In this mode, the negotiated keys are secure as long as at least one of the negotiated key exchanges are secure. There are two main design trade-offs as follows:

- **0-RTT support or not:** if supported (as depicted in Figure 7), this forces the client to send all key shares in its first message. Thus, increasing bandwidth requirements given the fact that most quantum-safe algorithms have relatively long keys. If 0-RTT is not supported and the design is derived from a TLS 1.3 handshake with mismatched parameters, then the first round trip will be used to negotiate the hybrid group and the second round trip to exchange only the key shares of the selected quantum-safe algorithms.

- **hybrid group definition and agreement:** the definition of hybrid groups also have pros and cons. One option consists in pre-defining and naming several hybrid groups so that client and server can just exchange their supported hybrid groups and pick up one of them. However, this might be complex due to the large range of candidates and configuration parameters. The alternative is that client and server agree on the hybrid group somehow, for instance, by exchanging their supported quantum-safe algorithms together with a policy that determines their requirements in the construction of a hybrid group.

Given this first overview of design trade-offs, the different specific proposals are detailed below.

The proposal by Whyte et al [i.12] supports a 0-RTT handshake and predefines hybrid groups. In this proposal, the TLS negotiation is essentially unchanged. The client issues an initial key exchange, which includes a list of supported groups and key shares for share material corresponding to zero (0) or more of the indicated supported groups. The server either selects one of the groups listed with a key share (and responds with its own key share), or it selects one of the groups listed as supported and issues a retry request listed within the selected group. The extension here is that the groups listed are not confined to be only DH or ECDH groups. They are also allowed to be either another key exchange or an indication of a hybrid group; that is, a combination of multiple specified key exchanges. The design puts no constraints on what groups can be included in the combination, except that each group appears no more than once, so the combination may, for example, be a single ECDH group and a single quantum-safe key exchange, or a combination of more than one quantum-safe key exchange, or some other combination type. For any hybrid group (that is, a logical group that is formed by running multiple key exchange mechanisms in parallel), the client will assign the named group id and its definition. Each individual key exchange mechanism has a defined key share format; this proposal also defines a format for key shares for the hybrid groups, designed so that even if two hybrid groups include the same key exchange mechanism, the key share material associated with that key exchange mechanism is only included in the handshake once.

The proposal by Schanck et al [i.22] does support 0-RTT and does not predefine hybrid groups. An extension "additional_key_share" is defined that allows the client to tell the server that it would like to negotiate an additional (quantum-safe) key share. If the server supports this extension, the server can then pick up one of them. The key schedule in the TLS 1.3 specification [i.12] is modified to incorporate the additional secret derived from the chosen quantum-safe key share. Note that if the client and server wish to agree on a hybrid group consisting of multiple quantum-safe algorithms (suppose N), they will have to repeat the above protocol N times to incorporate additional secrets derived from the N quantum-safe algorithms.

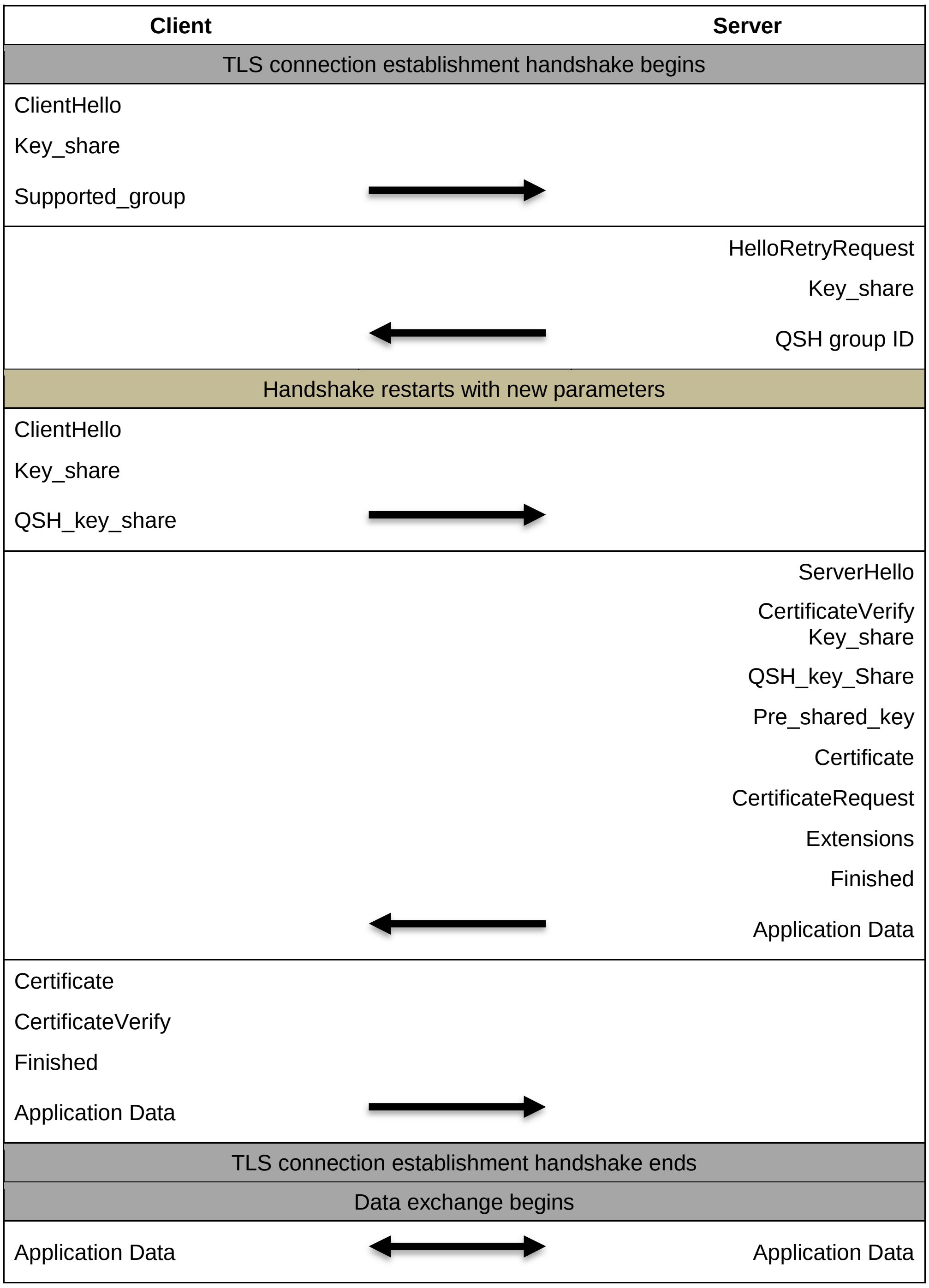

Figure 9: A Full Handshake with Mismatched QSH Parameters in TLS 1.3 [i.12]

6.3 Direct drop-in requirements

The main requirements for drop-in replacement are covered by the requirements presented in clause 6.2.1.

7 Media Access Control Security (MACsec) based requirements

7.1 Background

MACsec is defined in IEEE 802.1AE-2006 [i.24] and updated by IEEE 802.1AEbn-2011 [i.25], IEEE 802.1AEbw-2013 [i.26]. MACsec allows authorized systems in a network to establish confidential and authenticated connections for the transmission of data, commonly connecting systems to LANs or interconnecting LANs.

MACsec works by securing Ethernet frames between communicating nodes, typically a host to a switch/router or between two switches/routers.

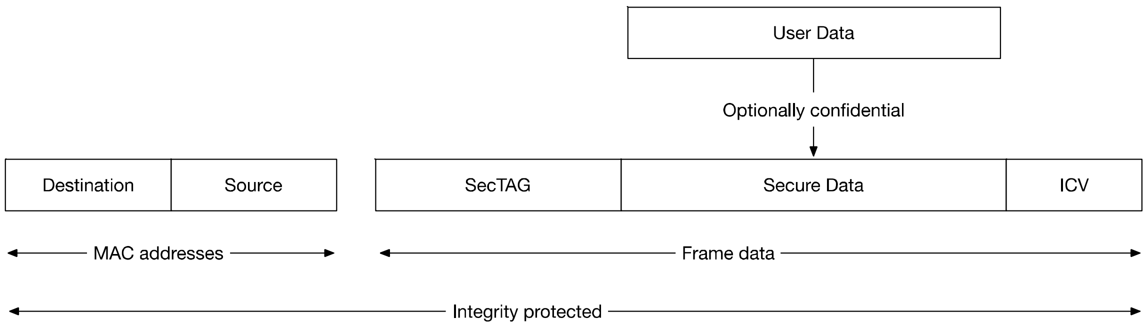

Figure 10: MACsec Frame

A connectivity association refers to the set of stations attached to a LAN that are mutually authenticated and authorized to establish a MACsec relationship to exchange protected data. A secure Connectivity Association Key (CAK) is a security relationship established by a key agreement method. A SA defines the security guarantees for the transmission of Ethernet frames between members of the connectivity association. Each SA is secured with the Security Association Key (SAK), forming a Secure Channel (SC).

The MACsec Key Agreement (MKA) protocol is responsible for discovering, authenticating, and authorizing the potential participants in a connectivity association. The protocol confirms possession of a secure CAK, and uses that CAK to establish a SAK. MKA is defined in IEEE 802.1X-2010 [i.27], and updated by IEEE 802.1Xbx [i.28], and IEEE 802.1Xck [i.29]. While the MKA protocol does more than establish fresh SAKs, this background is restricted to the cryptographic security of the distribution and agreement of keys.

The root in the key hierarchy for a MKA is the secure CAK. Each member of an SA will have possession of the CAK. The CAK is the result of an Extensible Authentication Protocol (EAP), a pre-shared key, or established by an MKA key server using an existing CAK, in practice. Every CAK has an associated secure Connectivity Association Key Name (CKN) that identifies the CAK.

When the CAK is established, one of the entities in the MKA protocol will be elected as the key server. The key server will generate a Key Encryption Key (KEK) and an Integrity Check Key (ICK) using a KDF. The KDF is defined in NIST SP800-108 as the counter Pseudorandom Function (PRF) where the PRF is AES-CMAC-128 (or AES-CMAC-256, based on the CAK being 128 or 256 bits).

The KDF can be considered as a single primitive that takes as input a cryptographic key (128 or 256 bits), a label, a key identifier, and an output length in bits (either 128 or 256 for a 128-bit or 256-bit output).

The KEK is then derived as KEK = KDF(CAK, "IEEE8021 KEK", CKN', [128, 256]) [i.28], where the CKN' is the first 16-bytes of the CKN, or zero-padded to 16 bytes.

Similarly, the ICK is derived as ICK = KDF(CAK, "IEEE8021 ICK", CKN', [128, 256]).

Each MKA Protocol Data Unit (MKPDU) is transmitted with a 128-bit integrity check value. The Integrity Check Value (ICV) value is the 128-bit AES-CMAC output of the message, where the message is destination address (DA) prepended to the source address and the MAC Service Data Unit up to but not including the ICV.

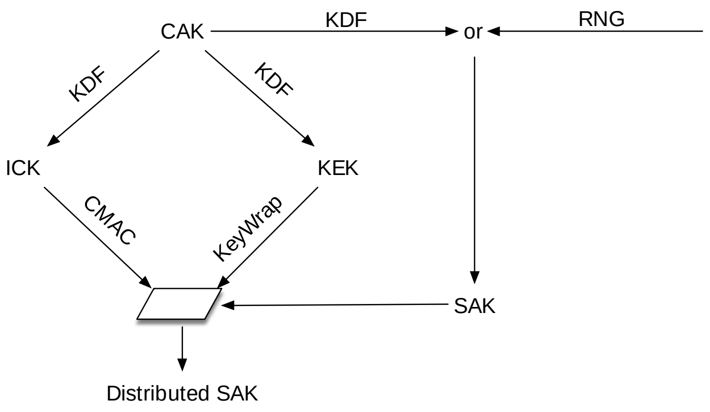

Figure 11: MKA Key Hierarchy

The key server can either generate a fresh 128 or 256-bit SAK, or use the KDF, to generate a new SAK. If the SAK is generated from the CAK it is computed using the same KDF as above as SAK = KDF(CAK, "IEEE8021 SAK", R, [128, 256]). R is constructed by using a nonce N (of the corresponding size 128 or 256), a list of Member Identifiers (MIs) and a Key Number (KN), four bytes assigned by the key server as part of the key identifier.

The SAK is encrypted using AES Key Wrap (AESKW) defined in IETF RFC 3394 [i.30].

The key server selects the cipher suite to be used. The cipher suites are defined IEEE 802.1AE-2006 [i.24], and updated by IEEE 802.1AEbn-2011 [i.25]. Currently they are restricted to AES-128-GCM and AES-256-GCM.

There is a similar scheme that can be used by a key server with CAKs established with entities which together can establish a group CAK. This is done by generating a group CAK and protecting under each pairwise CAK_i for each recipient via a separate ICK_i and KEK_i. Once established, a new secure connectivity association among holders of the group CAK can establish a SAK via the same process.

In addition to these methods, a CAK can be established as the result of an execution of an Extensible Authentication Protocol over LAN (EAPoL) method. An execution of EAP method results in a Master Session Key (MSK). In this case the pairwise CAK = KDF(Key, "IEEE8021 EAP CAK", M, [128, 256]). The key used is either MSK[0-15] or MSK[0-31] bytes depending on the CAK length of 128 or 256-bits respectively. The value M is the concatenation of the two authenticated entities MAC address used in the EAPoL-EAP exchange in numeric order (interpreted as an integer). Similarly, a secure Connectivity CKN is generated from the MSK, where CKN = KDF(Key, "IEEE8021 EAP CKN", M2, 128). The key uses are MSK[0-15] or MSK[0-31] depending on naming a 128 or 256-bit CAK key respectively. The value M2 is the EAP-Session-ID concatenated with the previously defined M value.

The EAP method specified in IEEE 802.1X-2010 [i.27] is the EAP (IETF RFC 3748 [i.31]), and further mandates the use of EAP-TLS (IETF RFC 5216 [i.32]) for the integration with IEEE 802.1AR [i.33]. EAP-TLS encapsulates a TLS record into EAP packets to perform a TLS negotiation. The MSK is derived from the master key between to the authenticating peer and the authenticator.

7.2 Hybrid requirements

On the surface, much of MACsec can be made quantum-safe by enforcing the use of 256-bit symmetric keys throughout the MKA protocol. The MSK, from which the CAK is derived in the EAP method may be made quantum-safe by mandating the use of a hybrid TLS cipher suite to establish the MSK.

MKA does not provide forward secrecy, which means a compromise of long term keys (CAKs) compromises past session keys (SAKs). This is a concern both from a classical and quantum perspective and is not specifically addressed in the present document.

7.3 Direct drop-in requirements

The root of the MACsec Key Agreement (MKA) protocol is a CAK. The CAK may be a pre-shared key, in which case no drop-in method is needed. However, when the CAK is transmitted or configured on the station, it typically uses another protocol, like the Simple Network Management Protocol (SNMP) or netconf, a network management protocol that runs over SSH. In either of these cases, a drop-in replacement to the underlying protocols for CAK establishment is needed.

In the case the CAK is established using a TLS-based EAP method, the underlying TLS ciphersuite would need to be augmented with a drop-in replacement solutions, see clause 6.3.

8 Secure Shell (SSH) based requirements

8.1 Background

The Secure Shell Protocol [i.3] provides services over a secure transport (most notably login shells, remote command execution, subsystem execution, and data stream forwarding). It natively supports TCP/IP port forwarding, and a common way of establishing layer 2 and 3 VPNs is to use the "tun@openssh.com" channel type defined by the OpenSSH project [i.4].

The protocol is of type client-server and is composed of three subprotocols. These subprotocols are not layered but multiplexed through the first byte of the packet payload, which acts as a message identifier. They are normally run in the following order:

- The TLP [i.5] provides confidentiality, integrity, and server host authentication. It defines a packet format (the Binary Packet Protocol) for transport of all Secure Shell data over TCP/IP. Key exchange, symmetric encryption, MAC, and public key algorithms are all extensible and negotiated. Once negotiated, confidentiality and integrity are applied to the payload of said packet for all subprotocols using keys derived from the key exchange.

- The User Authentication Protocol [i.6] provides user and client host authentication. The authentication method is extensible and the dialog can leverage public key algorithms.

- The Connection Protocol [i.7] exposes services via channels.

Figure 12 provides a simplified summary of the steps taken when establishing a VPN on top of a Secure Shell session, using the OpenSSH VPN extension as a channel example.

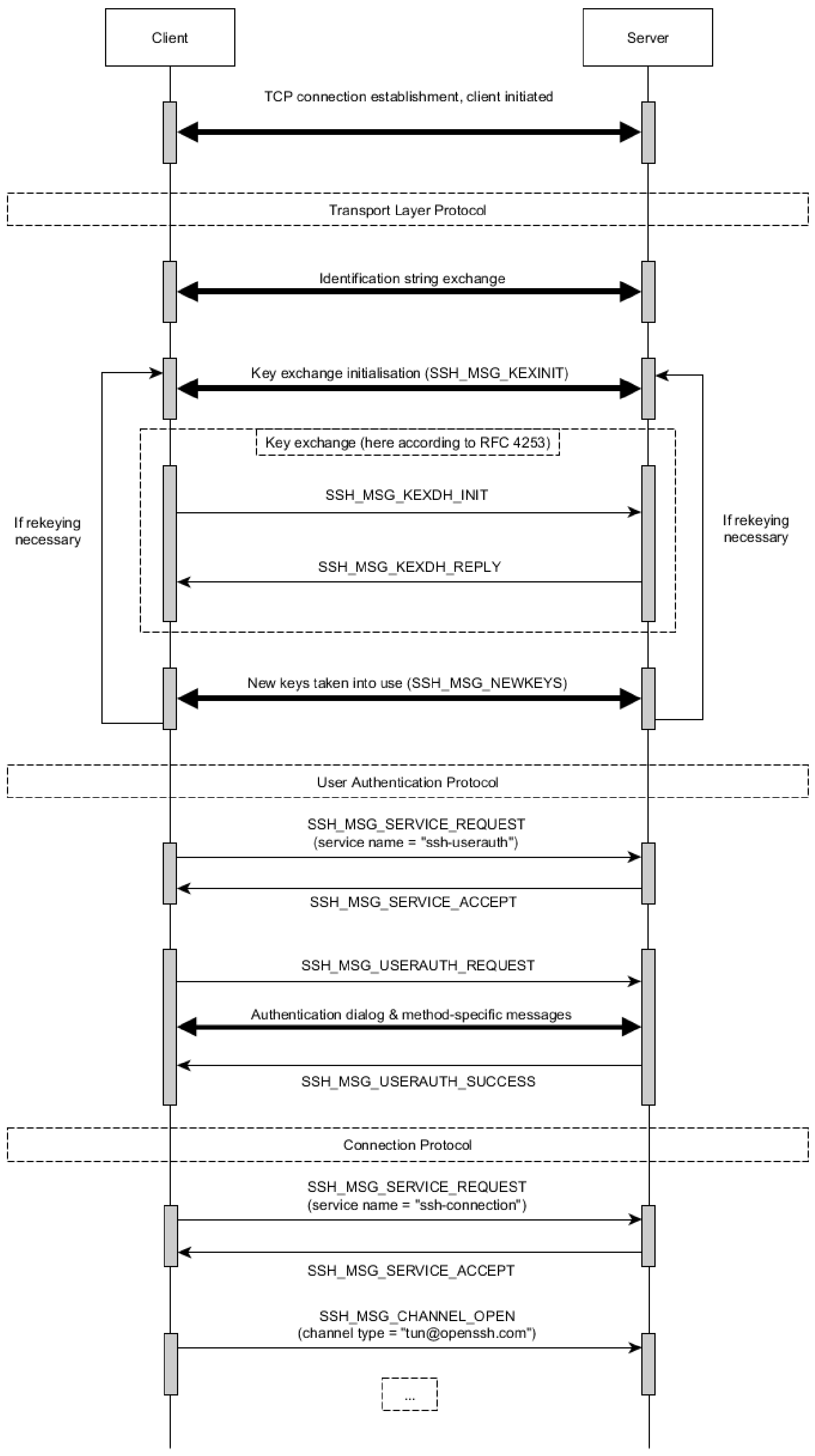
Client
Server
TCP connection establishment, client initiated
Transport Layer Protocol
Identification string exchange
Key exchange initialisation (SSH_MSG_KEXINIT)
Key exchange (here according to RFC 4253)
SSH_MSG_KEXDH_INIT
SSH_MSG_KEXDH_REPLY
If rekeying necessary
If rekeying necessary
New keys taken into use (SSH_MSG_NEWKEYS)
User Authentication Protocol
SSH_MSG_SERVICE_REQUEST (service name = "ssh-userauth")
SSH_MSG_SERVICE_ACCEPT
SSH_MSG_USERAUTH_REQUEST
Authentication dialog & method-specific messages
SSH_MSG_USERAUTH_SUCCESS
Connection Protocol
SSH_MSG_SERVICE_REQUEST (service name = "ssh-connection")
SSH_MSG_SERVICE_ACCEPT
SSH_MSG_CHANNEL_OPEN (channel type = "tun@openssh.com")
...

Figure 12: Simplified Example of a Secure Shell Session Establishment For a VPN

8.2 Analysis

While the Secure Shell Protocol is very flexible both from a cryptographic and a service extensibility perspective, there are some limitations to consider.

Several messages contain fields to transport public keys (either in the form of parameters or blobs) and digital signatures. Since the Secure Shell Protocol is expected to run on top of TCP [i.8] or a transport protocol with similar properties, reliable transmission of very large keys (such as those of code-based cryptosystems) is not a concern. However, support for such keys will require provisions in Secure Shell specifications. The Binary Packet Protocol [i.5] mandates that implementations be able to process Secure Shell packets of up to 35 000 bytes with an uncompressed payload of 32,768 bytes or less. The maximum packet size can be negotiated on a per-channel basis when a channel is opened, but this happens at the level of the Connection Protocol which comes too late in the session establishment to be of benefit to key exchange and public key algorithms. Similar concerns apply when quantum-safe digital signature algorithms producing very large signatures are used. This could be addressed with specific rules on packet size for implementations supporting quantum-safe key exchange methods and public key algorithms, but this approach is not flexible and could prove difficult to support for existing implementations. A more flexible approach would be to define message sequences adapted to the transfer of large blobs that would be split into chunks. Such rules would also apply to the Agent Protocol [i.9], which uses the Binary Packet Protocol.

In the sequel, hybrid modes are evaluated for both key exchange methods and public key algorithms (authentication).

As with other protocols studied in the present document, the time required to establish a Secure Shell session can increase due to the introduction of quantum-safe key exchange methods and public key algorithms, and will increase when hybrid modes are used for either of them. Secure Shell specifications have no notion of timeouts and the behaviour when dealing with long execution times (particularly on older hardware) solely depends on implementations. The performance of both TLP and the User Authentication Protocol are described in greater detail in the following two paragraphs.

TLP. TLP runs a key exchange algorithm, combined with a public key algorithm (for the server to prove its identity by means of a digital signature). These are negotiated separately during the initial handshake, and when rekeying is necessary ("kex_algorithms" and "server_host_key_algorithms" fields in the SSH_MSG_KEXINIT message as specified in [i.5], clause 7.1). While the rekeying operation does not discard in-flight messages, it does block any new message related to the User Authentication Protocol and the Connection Protocol until the rekeying is completed. Thus, the delay penalty of introducing hybrid modes is not limited to the initial session establishment but also applies during rekeying.

User Authentication Protocol. The User Authentication Protocol can involve the execution of one or more public key algorithms for the user and/or client to prove their identities with digital signatures. That is, the server can require one or more proofs based on public keys, for example one "hostbased" followed by one "publickey" authentication methods [i.6]. For each authentication method, the server can require that proofs be given with more than one (different) public key. Server configuration is the responsibility of the server administrator and is out of scope of the Secure Shell specifications. If hybrid authentication methods are introduced it can be very easy for the administrator to significantly increase the time required to complete the User Authentication Protocol.

The format of the SSH_MSG_KEXINIT message on the TLP limits the negotiation capabilities for hybrid modes. The design assumption is that the client and server will agree on one key exchange method and one public key algorithm (the latter for server authentication). This does not prevent the introduction of quantum-safe algorithms or hybrid modes, however for the hybrid modes it is not possible to separately negotiate the classic and quantum-safe algorithms that will be used. Several approaches to allow this would be possible:

- The format of the SSH_MSG_KEXINIT message could be modified to support separate algorithm negotiation. This message is not negotiated thus modifying it would require an upgrade to the protocol version number as the identification string is the only place where hosts can detect such change. It is questionable whether such drastic modification of the protocol is justified considering that hybrid modes will be used as a transition towards algorithms providing robust classic and quantum security.

- A new initialization message could be defined in the range 20 - 29 (for example, SSH_MSG_KEXINIT_HYBRID) that would support separate negotiation of classic and quantum-safe algorithms for hybrid modes. The specification would need to be modified for handling this new message (introducing unnecessary complexity) and it is not guaranteed that existing implementations will remain compatible.

- A new key exchange method could be defined, which would trigger additional steps in the key exchange procedure for negotiating classic and quantum-safe methods separately. This approach would require additional round-trips and render the key exchange inefficient.

Considering the disadvantages of all options above it seems better to stick to the behaviour of the existing specification and use composite names for key exchange methods and public key algorithm in hybrid modes. This will result in an increase of identifiers but this increase will remain within reasonable bounds. For example, as of drafting the present document, the OpenSSH server currently supports nine classic key exchange methods. If ten quantum-safe methods are considered for hybrid modes, this will amount to ninety possible combinations. However, not all combinations have to be accounted for in an extension to the Secure Shell Protocol specifications, and vendors can restrict their default configurations to enabling a handful of new key exchange methods (the most promising ones). If a set of quantum-safe key exchange methods is used in order to guard against progress in cryptanalysis against one method, making this set a fixed component in the method identifier will limit the number of possible combinations.

The handling of authentication messages in the TLP and the User Authentication Protocol depends on the way hybrid modes are defined for authentication. It is assumed that digital signatures will be applied side-by-side. In this case, messages holding digital signatures for authentication will need to have two or more signature fields - one for the classic algorithm and one or more for the quantum-safe algorithm(s).

Finally, the introduction of quantum-safe public key algorithms will also have an impact on the publication of key fingerprints in the Domain Name System using the SSHFP record [i.10]. While this is valid for any new public key algorithm, this matter is mentioned here for completeness.

8.3 Hybrid requirements

The following requirements are identified for key exchange methods and public key algorithms in hybrid mode:

- **Compliance with key exchange outputs**. All key exchange methods to output two values: a shared secret K, and an exchange hash H (as detailed in [i.5], clause 7.2). These are used in the derivation of initialization vectors, encryption keys and integrity keys. The hash H generated during the first key exchange is also used as a session identifier for other subprotocols, and is part of the data that is signed for authentication methods requiring public key algorithms. When hybrid key exchange methods produce the secret K and hash H in a final step, their impact on other aspect of the Secure Shell Protocol will be minimized. It is expected that hybrid key exchange methods will generate separate secret shares, using a specific method for each of them (e.g. a classic and a quantum-safe method). In that case, the shares are to be merged into the shared secret K, for example by using a KDF over the concatenated shares. Similar considerations may apply for the generation of the exchange hash H.

- **Fully specified hybrid mode names:**

    - When quantum-safe key exchange methods are used in conjunction with a classical method, specify new names for the kex_algorithms field in the SSH_MSG_KEXINIT message. For example, "alg1+alg2+alg3", where "alg1" is the name of a classical key exchange method, "alg2" and "alg3" are names of quantum-safe methods.

    - In order to limit the complexity of the key exchange phase (i.e. reduce the number of messages and avoid complicated negotiation), algorithms present in the specified hybrid mode names to remain uniquely ordered, and key exchange parameters to remain in the same order during message exchange. Similar requirements apply for public key algorithms used for the server host authentication in the server_host_key_algorithms field.

- **Definition of new key exchange messages and full key exchange specification:**

    - Define a new set of SSH_MSG_KEX*_INIT (message number 30) and SSH_MSG_KEX*_REPLY (message number 31) to be tied to a specific combination of hybrid mode name for key exchange and hybrid mode name for public key algorithm. The structure of the SSH_MSG_KEX*_INIT message is to accommodate the parameters of the first step of each key exchange algorithm (sent by the client host).

    - In case any of the key exchange algorithms require additional round-trips, use messages numbers in the range 32 - 49.

  - Structure of the SSH_MSG_KEX*_REPLY to accommodate the last step of each key exchange (generation of the shared secret K and exchange hash H, as above) as well as the server host authentication using a set of digital signatures.

- **Definition of new signatures for server host authentication**. When hybrid mode is used for server host authentication, the structure of the SSH_MSG_KEX*_REPLY to accommodate a set of tuples, each tuple being relevant to a specific public key algorithm and comprising a blob for the server host public key and a digital signature over the exchange hash H, for which the formats are eventually to be specified.

- **Efficient message exchange**. The creation of new key exchange methods for hybrid modes will lead to the definition of new messages in the range 30 - 49, semantically linked to the execution of these methods. In order to limit the number of round-trips between the hosts and thus limit the effect of network latency, specify messages such that key exchanges can be run in parallel instead of being serialized, i.e. each message carries sections relevant to each method. In the ideal case, this can reduce the overall process to one round-trip provided each exchange method requires one round-trip only.

- **Large messages:**

  - When the cumulated size of blobs in a given message is expected to result in an overall uncompressed payload of more than 32,768 bytes for the underlying Binary Packet Protocol packet, specify the message sequence (key exchange or public key authentication) to accommodate transfer of the blob(s) in chunks.

  - Specify one or more new message types in the range 32 - 49 as intermediary messages between SSH_MSG_KEX*_INIT and SSH_MSG_KEX*_REPLY messages.

  - Use the range 54 - 59 in the User Authentication Protocol to define messages supporting transfer of blobs in chunks.

NOTE: This requirement complements those defined previously in the present clause.

The following requirements are identified for public key algorithms in hybrid mode, at the level of the User Authentication Protocol:

- **Serialized authentication**. Reuse the SSH_MSG_USERAUTH_REQUEST and SSH_MSG_USERAUTH_PK_OK messages to support hybrid mode with one or more quantum-safe public key algorithm, provided all requirements defined in clause 8.3 for the User Authentication Protocol are followed. This is because the authentication requests can be serialized, with each request focusing on one authentication method or public key algorithm.

8.4 Direct drop-in requirements

The following requirements are identified for the key exchange and public key algorithms, at the level of the TLP:

- **Definition of new key exchange messages:**

  - When a quantum-safe key exchange algorithm is used, specify a new name (for the kex_algorithms field).

  - Specify a new set of SSH_MSG_KEX*_INIT (message number 30) and SSH_MSG_KEX*_REPLY (message number 31) tied to the quantum-safe key exchange algorithm name.

  - Structure of the SSH_MSG_KEX*_INIT message to accommodate the parameters of the first step of the key exchange algorithm (sent by the client host).

  - In case the key exchange requires additional round-trips, use messages numbers in the range 32 - 49.

  - Structure of the SSH_MSG_KEX*_REPLY to accommodate the last step of the key exchange (generation of the shared secret K and exchange hash H, as in clause 8.2 above) as well as the server host authentication using a single digital signature.

- **Definition of a new signature for server host authentication:**

  - When a quantum-safe public key algorithm is used for server host authentication, specify a new name (for the server_host_key_algorithms field).

  - Structure of the SSH_MSG_KEX*_REPLY to accommodate a blob for the server host public key and a digital signature over the exchange hash H, for which the formats are to be specified.

- **Full key exchange specification:**
  - Specify the possible combinations of the key exchange and public key algorithms so that implementations can conclude algorithm negotiation and process SSH_MSG_KEX*_INIT and SSH_MSG_KEX*_REPLY messages.

The following requirements are identified for public key algorithms, at the level of the User Authentication Protocol:

- **Reuse of existing message format:**
  - For the "publickey" and "hostbased" authentication methods, reuse the SSH_MSG_USERAUTH_REQUEST and SSH_MSG_USERAUTH_PK_OK messages as they are currently defined in clause 7 of [i.6], and clause 9 of [i.6], provided a name is defined for the quantum-safe public key algorithm (to be used in the "public key algorithm name" or "public key algorithm for host key" fields) and the format of the public key blob and signature blob for that algorithm name are specified.
  - Follow the rules of signature generation (e.g. the use of the exchange hash H as session identifier).

The following generic requirement is identified:

- **Large messages:**
  - When the cumulated size of blobs in a given message is expected to result in an overall uncompressed payload of more than 32,768 bytes for the underlying Binary Packet Protocol packet, specify the message sequence (key exchange or public key authentication) to accommodate transfer of the blob(s) in chunks. For example, one or more new message types in the range 32 - 49 could be defined as intermediary messages between SSH_MSG_KEX*_INIT and SSH_MSG_KEX*_REPLY messages.
  - Use the range 54 - 59 in the User Authentication Protocol to define messages supporting transfer of blobs in chunks.

NOTE: This requirement complements those defined previously in the present clause.

9 Conclusion

Over the course of the present document, a number of the requirements, and potential solutions were explored, to get VPN infrastructures ready for quantum-safety. There have been a number of consistent requirements that have come up in both the hybrid and direct drop-in options. These have included a focus on backwards compatibility, limiting data exchanges when appropriate, and limiting complexity when possible. Ultimately, the observation is that this transition is very complex and organizations should take this as a reason to start migration planning early to ease this transition, and minimize costs and disruption to their business.

Annex A: Experimental Results Related to Message Fragmentation

In the paper "The Viability of Post-Quantum X.509 Certificates", [i.18] Kampanakis et al. discuss a number of the challenges associated with utilizing new Quantum-Safe algorithms within existing protocols such as X.509 and TLS and IKEv2.

The main area that they focus is related to implications due to the increase in packet sizes and object sizes because of larger public key and signatures. They do not focus on issues related to increased processing requirements for the new schemes.

What they found is that large Quantum-Safe certificate chains can be managed in common Internet protocols through the use of segmentation and fragmentation. This increases overhead but not in an unacceptable fashion. Their results suggest that a transition to Quantum-Safe algorithms should be successful but that testing from a performance/size and backwards and forwards compatibility will be key.

Annex B: Additional Protocol Impacts

During the analysis of Quantum-Safe algorithm options, a number of new issues are being discovered that are due to the nature of the schemes themselves. These properties can impact protocol design and so should be noted.

Initiator/Responder Paradigm

A number of Quantum-Safe key establishment schemes are built around the idea of an "initiator" and "responder". The result of this design is that the keying material contribution from the responder incorporates aspects of the material produced by the initiator in such a way that precomputation can be difficult. This is in contrast to ECDH where both parties can precompute tables of information to speed up the key establishment process.

Using a Key Encapsulation Mechanism **(KEM) for Key Transport**

Quantum-safe key transport schemes are generally KEMs. This is due to the fact that there are padding-related attacks on quantum-safe schemes. In a KEM, an encrypted seed is transmitted where the size of the seed is such that it does not require padding. When using a KEM to achieve key transport, the dependent protocol needs to be modified because KEM is responsible for generating the keying material from this seed. This should be a relatively straight forward change, but without it a direct drop-in replacement will not work.

Hybrid Key Transport

In a hybrid key agreement, peers can negotiate multiple shared secrets that can be combined before symmetric keys are generated. If multiple quantum-safe schemes (based on different hard math problems are used for greater confidence in the overall strength) are used, they can all be combined into one shared secret before the symmetric keys are generated from it. This is not that easy to do in key transport. While classic schemes are public key encryption schemes and can take any arbitrary plain text, quantum-safe schemes are all KEMs that cannot do that. If one wanted to achieve hybrid key transport by double encryption, it can only work if the classic scheme is used to encrypt the KEM's ciphertext. Doing it in reverse would not work, because KEM does not take arbitrary plaintext. This also becomes a challenge if more than one quantum-safe scheme (or KEM in this case) based on different hard math problems is used. If multiple quantum-safe schemes need to be or the KEM needs to be on the "outside", intermediate symmetric key needs to be generated that would encrypt the arbitrary plaintext. The seed the symmetric key generated from would be encapsulated by KEM.

Static Keys

When using quantum-safe schemes for key establishment, some constructions are only safe in ephemeral-ephemeral modes as they allow attacks when used in static modes. For example, [i.19] Gailbraith et. al. discuss a method where an attacker can learn about the bits of the peer's private key. In this scenario, the attacker uses a different key pair, while the victim keeps using the same key pair. This makes a static-ephemeral DH scenario insecure. Therefore, the use of SIDH in the one-pass protocol, where a public key encryption instance is created, should be avoided. Fully ephemeral key agreement remains secure. This means the protocol properties need to be carefully considered when choosing a quantum-safe scheme to utilize.

Mathematical Basis for Hybrid Schemes

Security, authenticity and forward secrecy against classical computers are inherent from the classical handshake mechanism. In this transition period as the new Quantum-Safe algorithms are standardized, there is a desire to keep the properties offered by existing classical handshakes but adding protection from quantum computers. As a result, the use of a hybrid scheme then provides quantum security and quantum forward secrecy while maintaining the classical handshake properties.

Mathematically, this security can be seen with the Leftover Hash Lemma (LHL). Suppose that the Quantum Safe Hybrid (QSH) scheme has keying material S. Given an input X, the LHL ensures that one can extract Y bits that are almost uniformly distributed, where Y is asymptotic to the min-entropy of X. An adversary who has some partial knowledge about X, such as the classical contribution, will have almost no knowledge about Y. This guarantees the attacker will not learn the final premaster secret so long as S has enough entropy and remains secret. This also guarantees the premaster secret is secure even if the client's and/or the server's long-term keys are compromised.

History

Document history		
V1.1.1	September 2018	Publication