Sichere Integration von QKD-Keys

Jasmin Neumann, Vincent Burkard

Inhalt

Einführung zur sicheren QKD-Schlüsselintegration

Der Vorteil von Quantum Key Distribution (QKD) ist informationstheoretisch sichere Kommunikation auch in der Zukunft, die unbemerktes Abhören bei richtiger Implementierung vollständig eliminieren kann. Gerade für Behörden, Banken und andere sensible Infrastrukturen ist dies von zentraler Bedeutung.

In Anbetracht der zunehmenden kommerziellen Verfügbarkeit von QKD-Geräten auf dem Markt ist zu überlegen, wie diese QKD-Schlüssel geeignet in bestehende Protokolle integriert werden können. Die Standardisierung zieht sich hin (z. B. Veröffentlichung des Inter- *Key Management System* (KMS)-Standards ETSI 020 nicht mehr 2025), während die praktische Anwendung der quantensicheren Protokolle spätestens bis 2030 erfolgt sein sollte, da die klassischen Verschlüsselungen brechenden Quantencomputer hypothetisch ab 2030 verfügbar sein könnten [1]. Auch die *Agenda Quantensysteme 2030* fordert, dass die Entwicklung von QKD zunehmend in klassische Protokolle miteingebunden wird: vom kurzfristigen Ziel einer abhörsicheren, kurzen Verbindung der Gerätepaare über größere Quantennetze bis hin zum erklärten Langzeitziel: einem globalen Quantennetz [2]. Deshalb widmet sich dieses Dokument schon jetzt der Frage, wie QKD-Schlüssel in klassischen Netzwerken zu deren Absicherung eingesetzt werden können.

Teil A dieses Dokuments (siehe Abbildung 1) gibt zunächst einen Überblick über eine Reihe von Sicherheitsprotokollen, wie sie auf den verschiedenen Ebenen (Layern) des ISO/OSI Schichtenmodells im Moment eingesetzt werden und untersucht, inwieweit sich eine Integration der QKD-Schlüssel anbieten und eignen würde. Erklärt wird für das jeweilige Sicherheitsprotokoll welche Änderungen vorzunehmen sind und wie der Integrationsaufwand/-machbarkeit eingeschätzt werden.

In Teil B werden quantensichere Authentifizierungsverfahren genauer betrachtet. Authentifizierungsverfahren sind eng verknüpft mit Sicherheitsprotokollen und QKD, da bei einem Schlüsselaustausch gesichert klar sein muss, wem der Schlüssel zur Verfügung gestellt wird. In diesem Kapitel wird bewertet, ob ein klassischer Nachrichtenauthentifizierungscode (MAC) in Bezug auf die Quantensicherheit genügt und ob die quantenbasierten Verfahren zur Authentifizierung schon Potential für die Praxis besitzen.

Das letzte Kapitel C untersucht Hybridisierung mit QKD, d.h. wie QKD-Schlüssel sich zum Kombinieren mit anderen Schlüsseln aus klassischen Verfahren eignen, oder sich in Kombination mit PQC (Post-Quantum Cryptography) einsetzen lassen. Auch hybride Methoden zur Authentifizierung zusammen mit QKD werden in diesem Teil abschließend diskutiert.

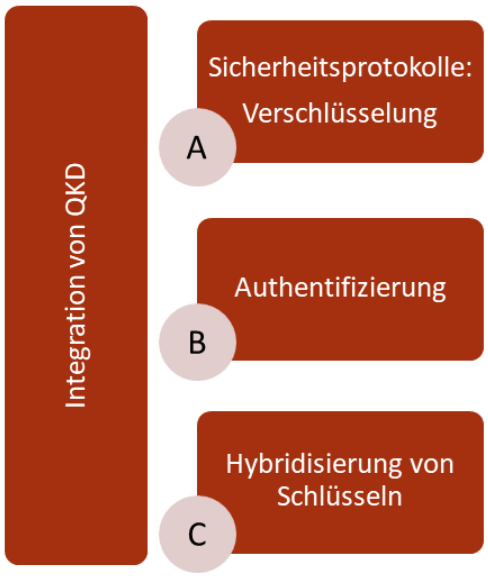


Abbildung 1: Übersicht über die Einsatzmöglichkeiten von QKD in der Netzwerksicherheit für eine andauernde Resistenz gegenüber Quantencomputern: QKD bei Sicherheitsprotokollen mit Fokus auf Verschlüsselungsprotokolle, bei Authentifizierung extra und bei der Hybridisierung von Schlüsseln.

A. Einsatz der QKD-Keys in verschiedenen Sicherheitsprotokollen

Das International Organization for Standardization/ Open Systems Interconnection (ISO/OSI) - Referenzmodell [3] beschreibt die Funktionen der Netzwerkkommunikation mit Hilfe von sieben Schichten und stellt dar, wie Standard- und Sicherheits-protokolle auf diesen Schichten eingesetzt werden um Netzkommunikation zu regeln.

Im Folgenden wird die Integration von QKD-Keys in Sicherheitsprotokolle unterschiedlicher Schichten dieses ISO/OSI-Referenzmodells betrachtet. Dabei wird vorausgesetzt, dass die QKD Schlüssel von den QKD Geräten generiert werden, und über das Schlüsselmanagementsystem für einen Einsatz in den jeweiligen Protokollen zur Verfügung stehen.

Sicherheitsprotokolle definieren neben den Verschlüsselungsmethoden meist auch die Authentifizierungsmethoden; besonders in den Protokollen höherer Schichten können deswegen Sicherheitsprotokolle und Authentifizierung nicht unabhängig voneinander betrachtet werden.

Während es bei den Verschlüsselungsmethoden meist problemlos möglich ist, kleine Anpassungen, wie Schlüssellänge, kryptographische Parameter oder die Quelle statischer Schlüssel abzuändern, stellt eine Integration von dynamischen QKD-Keys eine tiefgreifende Abänderung der bestehenden Sicherheitsprotokolle dar. Es gibt bisher nur sehr wenige Protokolle, die es erlauben, QKD-Schlüssel zu integrieren. Einige Protokolle könnten mit geringen Abänderungen für QKD angepasst werden. Üblicherweise bringen die Protokolle aber ihre eigenen selbst-verhandelnden asymmetrischen Schlüsselaustauschmechanismen mit, welche komplett durch symmetrische QKD ersetzt werden müssten. Die in den Protokollen eingesetzten Verschlüsselungsmechanismen (häufig Advanced Encryption Standard (AES)[1] oder One-Time-Pad (OTP)) könnten an sich einfach zur Verfügung stehende QKD-Schlüssel zur Enkryption verwenden, wenn sie zuvor in der Schlüsselaustauschfunktion im Protokoll definiert wurden und diese durch ein externes Schlüsselnetzwerk von speziellen QKD-Geräten als Key-Server bereitgestellt werden.

Auch die darin teils fest verankerten Authentifizierungsmethoden sollten idealerweise durch quanten-sichere Methoden ersetzt werden, wobei diese Anforderung nur zur Laufzeit gelten muss: Ein informationstheoretisch sicherer Message Authentication Code (MAC) sollte im Protokoll verwendet werden, um die Authentizität und Integrität der Nachricht während der Übertragung sicherzustellen (weiteres in Kapitel B).

Für den Einsatz von QKD-Keys sind dynamisch ersetzbare Pre-Shared Keys (PSKs) *Out-of-band* nötig und ein sicherer, authentifizierter Schlüsselkanal als dauerhaft definierte, externe Quelle. Dazu gibt es bereits u.a. von ETSI die Standards 004/014 [4], [5], die speziell für den sicheren Austausch der QKD-Schlüssel mit Enkryptoren entwickelt wurden. Unter der Voraussetzung, dass ein Verschlüsselungsprotokoll externe PSKs unterstützt, wird nur für den Austausch der QKD-Schlüssel gesorgt, aber eine Integrierung in klassische Sicherheitsprotokolle von dynamischen QKD-Schlüsseln, ist bisher nicht direkt standardisiert. Hier ist es für die Praktikabilität nötig, die bisherigen Standards für die Verwendung mit QKD-Keys zu erweitern. Dies bringt aber weitere Probleme für die Protokolle z.B. in der Kompatibilität, in der Performance und den zur Verfügung stehenden Raten mit sich.

[1] AES-256 kann auch in Zukunft noch als quantensicher angesehen werden: Die Autoren in [4] haben gezeigt, dass Schwierigkeitsgrad für das Aushebeln von AES-128 auf herkömmlichen Computern dem Schwierigkeitsgrad für das Kompromittieren von AES-256 auf Quantencomputern entspricht.

Schlüsselströme, die von QKD-Geräten bereitgestellt werden, sind jedoch nicht immer synchron abrufbar und im passenden Format für die direkte Verschlüsselung vorhanden. Bei kommerziellen Switchen/Routern/Firewalls wird nun auch zunehmend eine Integration von Quantenschlüsseln in verschiedene Protokolle angeboten. Eine generelle Erweiterung der Protokoll-Standards für QKD-Keys wäre unbedingt nötig, um nicht auf proprietäre Protokolle einzelner Hersteller zurückgreifen zu müssen.

Bei der Wahl der jeweiligen Schicht zur Verschlüsselung ist man grundsätzlich frei und sie hängt meist von der jeweiligen Anwendung ab. Dabei gilt generell: je weiter unten der Layer, desto weniger Overhead und damit Latenz, da die verschlüsselten Pakete jeweils für die höheren Layer transparent sind. Somit gilt die Verschlüsselung auf der physikalischen Schicht (Layer 1) als besonders sicher. Abbildung 2 zeigt eine Übersicht über mögliche Verschlüsselungen mit QKD-Keys der Sicherheitsprotokolle auf unterschiedlichen Schichten, wie sie im Folgenden diskutiert werden.

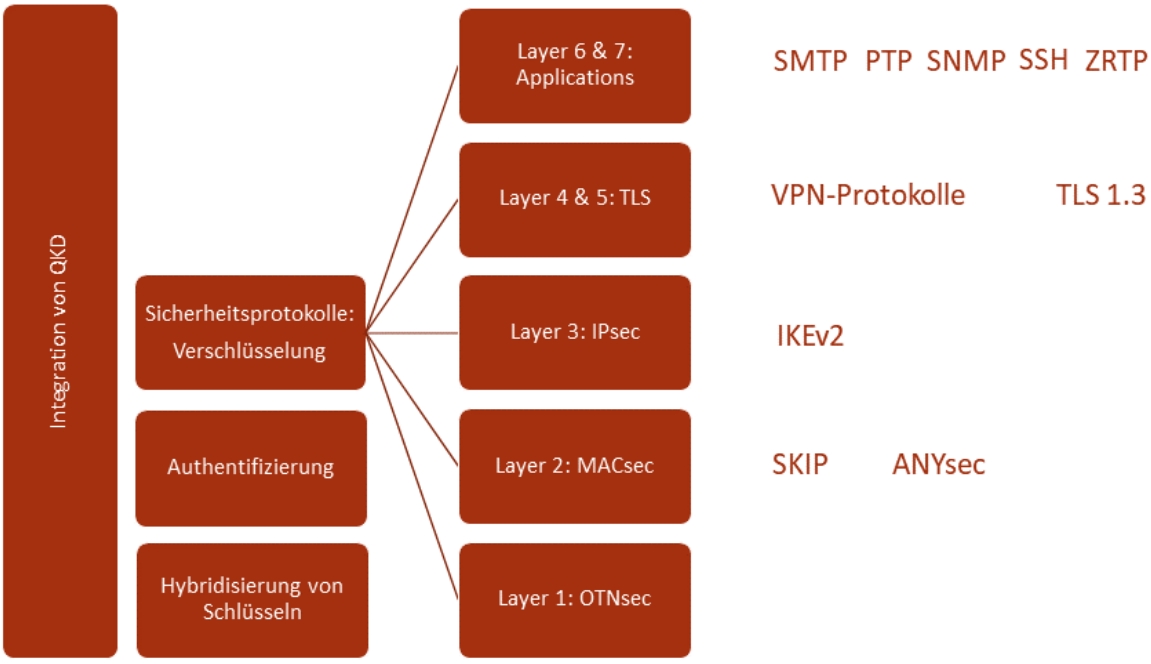


Abbildung 2: Übersicht Kapitel A: Sicherheitsprotokolle und deren Verschlüsselungen auf verschiedenen Schichten: 1. OTNsec, 2. MACsec, 3.IPsec, 4./5. TLS, 6./7 SMTP, PTP, SNMP, SSH, ZRTP.

Schicht 1

Das Optical Transportnet Protocol wurde von der ITU v.a. in G.709.1-6, G.959.1 (Schnittstellen), G.872 (Architektur) & G.798 (funktionelle Anforderungen Equipment) beschrieben. Das OTN besteht im Wesentlichen aus einer optischen Transmissionssektion (OTS), einer optischen Multiplexsektion (OMS) und einem optischen Kanal (OCh). Zudem wird Wellenlängenmultiplexing (WDM) und Forward Error Correction (FEC) ermöglicht.

Der OTNsec-Standard der ITU, der sich mit der Sicherheit von OTN befasst, ist leider noch nicht öffentlich verfügbar [6]. Das Protokoll OTNsec verschlüsselt auf der physikalischen Schicht und ist somit besonders sicher, da die verschlüsselten Bits auch auf den höheren Ebenen automatisch verschlüsselt sind bei unautorisiertem Zugriff. Außerdem können mit OTNsec Datenraten von mehr als 100 Gbit/s erreicht werden. Zudem ist alles, inklusive der Steuerdaten im Header automatisch verschlüsselt, statt nur der Nutzdaten, wie bei vielen Protokollen auf höheren Ebenen. Es können gewöhnliche Verschlüsselungsmethoden, häufig AES-256, damit verwendet werden. Zur Verhandlung über das verwendete Verschlüsselungsverfahren kann auch IKEv2 (siehe Schicht 3) [7] eingesetzt werden. OTNsec genießt generell weniger Aufmerksamkeit, wurde aber schon erfolgreich mit QKD-Schlüsseln kombiniert [8], [9], [10].

Die Autoren Makris et al [8] haben zum Beispiel L1-OTNsec-Verschlüsselung in Verbindung mit QKD Schlüsseln in einem Netz mit zwei Endknoten Alice und Bob und einem Trusted Node dazwischen demonstriert. Dabei kam ein Hybridschema aus QKD und klassischen quantensicheren zentral generierten symmetrischen Schlüsseln zum Einsatz. Abbildung 3 zeigt den Einsatz der Layer1-Enkryptoren an den drei Knoten auf der Teststrecke. Das Netzwerk besitzt neben dem klassischen QKD-Service-Channel 30 die OTNsec-Verbindungen des Channels 34, 36 und 38. Jeder Knoten verfügte für Anwendungen über einen Nokia Layer 1/OTN Photonic Service Interconnection-Modular (PSI-M) [11] Verschlüsselungsprozessor (3x100 Gbps) zur Verschlüsselung von Daten mit QKD. Es wurden dabei verschiedene Schlüsselrotationsraten von einem Schlüssel pro Minute bis hin zu einer Stunde verwendet. Die PSI-M Enkryptoren 1&2 waren miteinander über die WDM-Strecke von 45 km verbunden, welche durch zwei Zirkulatoren bidirektional genutzt werden konnte. Das OTNsec-Signal von Channel 36 wurde mit dem Channel 34 und dem Service-Channel 30 des ersten QKD-Gerätepaares gemultiplext. Die jeweils anderen zwei Verbindungen sind mit zwei Dark Fibers über 13km bzw. 45km Strecken realisiert worden. Zusätzlich wurde ein Nokia 1830 Security Management Server (SMS) als Sicherheitsorchestrator eingesetzt, der gleichzeitig die QKD-Schlüsselverteilung über die Knoten des Netzwerks und auch die Schlüsselanforderungen der klassischen OTN-Schicht orchestrierte und bei einem Angriff auf die QKD-Schicht in der Lage war, nahtlos zu einer klassischen quantensicheren Verschlüsselung überzugehen.

In einer ähnlichen Kombination von OTNsec und QKD haben die Autoren Pincemin et al. in [9] den einen 100 Gbps Transponder (Adva FSP 3000, vom Bundesamt für Sicherheit in der Informationstechnik (BSI) zugelassen [12]) zur Verschlüsselung des Datenstrom mit AES via OTNsec verwendet. Zusätzliche 4x100 Gbps Muxponder mit 100GBASE-LR4 QSFP28 Interface, erzeugten die insgesamt 400 Gbps Dual-Polarization 16 QAM Kanäle. Diese konnten je einen OTNsec 100 GbpsEthernet QKD-verschlüsselten Datenstrom der Transponder tragen und mit anderen drei 100 Gbps (hier unverschlüsselten) Datenströmen kombiniert werden. Das WDM enthielt im Experiment jeweils 54 Wellenlängen. Dabei waren die einzelnen Faserabschnitte zwischen 50-67 km lang.

Da es bisher kaum dedizierte Glasfasernetze für QKD gibt, muss bei einem Einsatz von QKD auf der physikalischen Schicht generell beachtet werden, dass QKD spezielle Anforderungen an die Faserinfrastruktur mitbringt. Verstärker auf der Strecke können die schwachen Photonen auf der QKD-

Strecke zum Beispiel zerstören [9]. Außerdem sind die Reichweiten von QKD Systemen und WDM Systemen unterschiedlich, d.h. im QKD-Netzwerk (QKDN) muss spätestens alle 150 km (abhängig vom jeweiligen QKD-Gerät) das Signal an sog. *Trusted Nodes* neu verschlüsselt werden und es werden Key Management Systeme (KMSs) benötigt, die die Schlüssel verwalten.

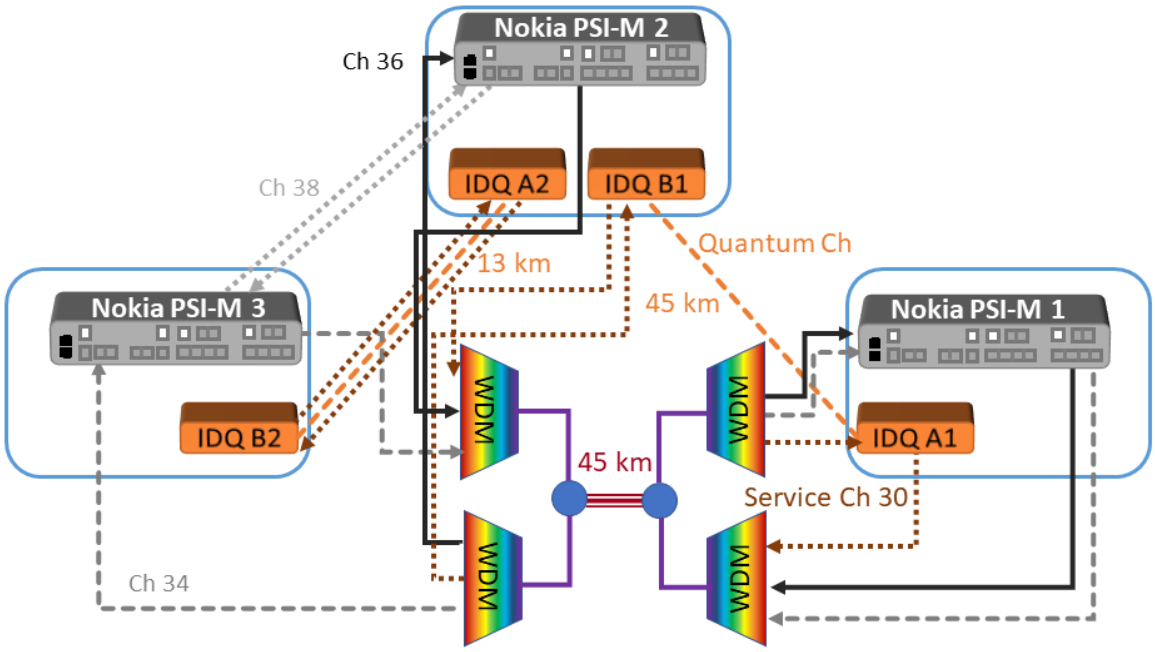


Abbildung 3: Beispiel für OTNsec über 45 km WDM mit drei Nokia PSI-M Enkryptoren und drei Channeln dazwischen: Ch 36 (OTNsec PSI-M 1-2) schwarz/durchgezogen, Ch 34 (OTNsec PSI-M 1-3) mittelgrau/gestrichelt und Ch 38 hellgrau/gepunktet (OTNsec PSI-M 2-3); Service-Channel (Ch 30) braun/gepunktet und Quantum Channel orange gestrichelt zwischen den IDQ-QKD-Geräten, dabei wurden Ch 34 und Ch 36 mit dem Service-Channel 30 des ersten QKD-Gerätepaares WDM - gemultiplext. [8]

Schicht 2

MACsec

Media Access Control security (MACsec) bietet sowohl eine Verschlüsselung der Nutzdaten (via AES), als auch eine Authentifizierung (siehe Abbildung 4). Vor dem Header wird, wie für einen Ethernetframe üblich, eine *Destination Medium Access Control* (DMAC)- und eine Source MAC-Adresse definiert. Der Header enthält den MAC Security TAG (SecTAG) mit dem MACsec Ether Type, der TAG Control Information (TCI), der Association Number (AN), der Shorth Length (SL) und der Packet Number (PN), welche gegen Replaying schützt und bis 2^{64} in der Extended (XPN) gehen kann. Auch ein optionaler Secure Channel Identifier (SCI) ist im MACsec Header vorgesehen. Die Payload kann encodiert werden, ebenso wie der SecTAG, je nachdem wie die Managementbits gesetzt sind. Cipher Suites sind AES-GCM(-XPN)-128/256. Zusätzlich sieht das Protokoll einen Integrity Check Value (ICV) vor dem zyklischen Redundanzcheck (CRC) vor. [13]

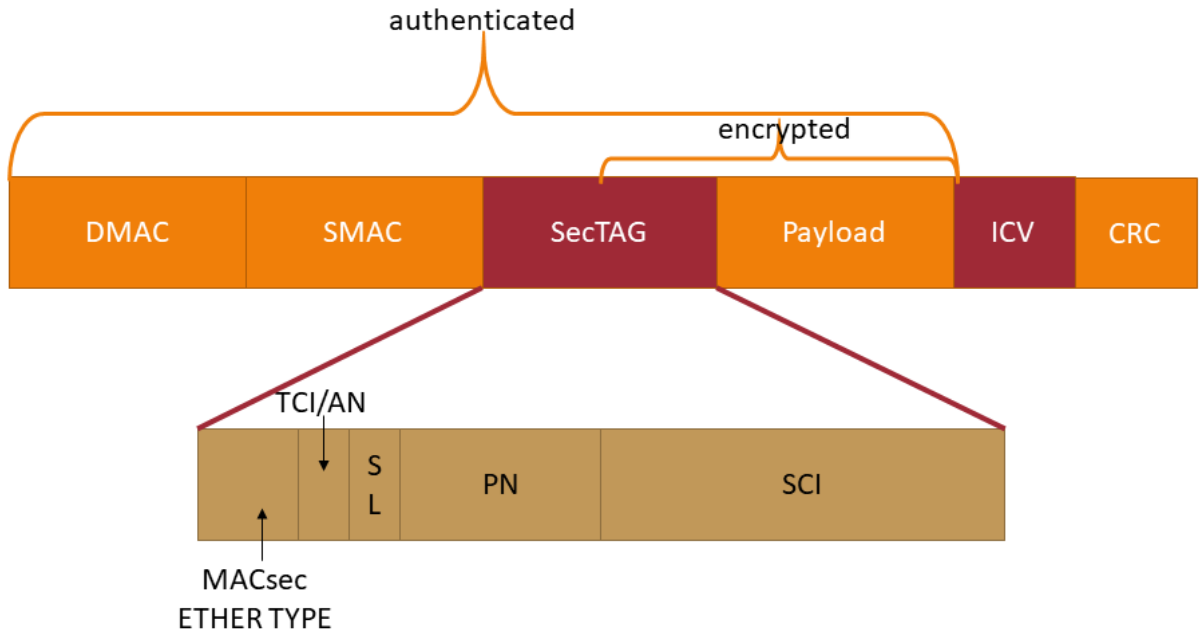
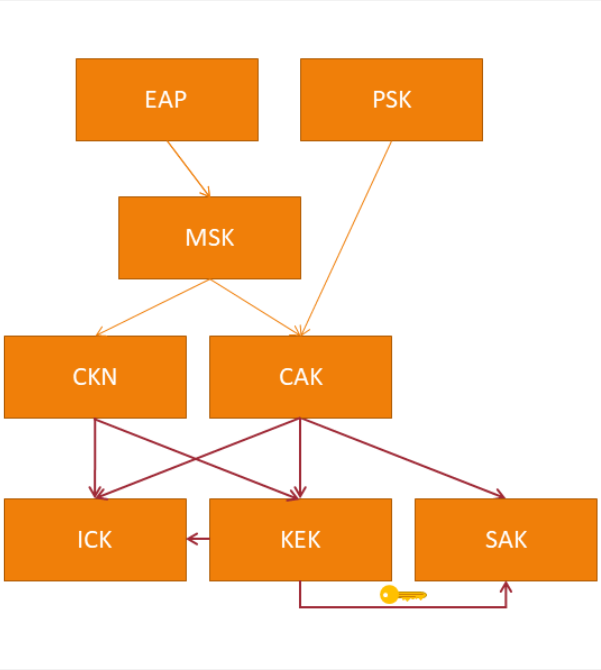


Abbildung 4: Die Paketierung von MACsec: Header mit Destination Medium Access Control (DMAC)- und eine Source MAC-Adresse, MAC Security TAG (SecTAG) mit dem MACsec Ether Type, der TAG Control Information (TCI), der Association Number (AN), der Shorth Length (SL) und der Packet Number (PN), optionaler Secure Channel Identifier (SCI), Integrity Check Value (ICV), zyklischen Redundanzcheck (CRC).

MACsec besteht standardmäßig aus einer Data Plane (IEEE 802.1AE [13]: Definition der Header und verschlüsselten Paketformaten) und einer Control Plane (IEEE 802.1X [14]: Erzeugung und Verteilung der Schlüssel (*MACsec Key Agreement* (MKA)); klassisch wird das Extensible Authentication Protocol (EAP) verwendet). Es werden zunächst durch den Nutzer Secure *Connectivity Associations* (CAs) definiert; ausgehend von einem Pre-Shared *Connectivity Association Key* (CAK)/Master Key, der pseudo-statisch ist und nicht oft geändert wird, wird ein häufig zu aktualisierender *Session Association Key* (SAK) erzeugt und verschlüsselt an die berechtigten Partner weitergeleitet. Während der CAK nie

übersendet wird, wird der SAK ständig über den gemeinsamen Link geteilt. Die Zusammensetzung der Keys aus dem CAK ist in Abbildung 5 veranschaulicht mit einer Übersicht über die Abkürzungen.



AES	Advanced Encryption Standard
CAK	Connectivity Assosciation Key
CKN	Connectivity Association Key Name
EAP	Extensible Authentication Protocol
GCM	Galois Counter Mode
ICV/ICK	Integrity Check Value/Key
KDF	Key Derivation Function
KEK	Key Encrypting Key
MKA	MACsec Key Agreement
MSK	Master Session Key
PSK	Pre-Shared Key
SAK	Secure Association Key

Abbildung 5: Die verschiedenen Keys und deren hierarchische Ableitung beim MACsec Key Agreement (MKA): Aus EAP wird MSK, woraus man CKN und mit PSK den CAK erhält. Aus dem KEK kann SAK und zusammen mit CKN der ICK bestimmt werden.

Es ist grundsätzlich möglich, sowohl den CAK als auch die SAKs über QKD bereitzustellen, wie im Folgenden eingehend erläutert wird:

Bei MKA gibt es grundsätzlich zwei Methoden, den CAK zu erhalten (vgl.

Abbildung 5): durch die Verwendung eines MSK vom Extensible Authentication Protocol (EAP) und mit einem PSK. Der MSK, der aus EAP abgeleitet wird, welches von der Public Key Infrastructure (PKI) abhängt, kann einfach durch einen QKD-Key ersetzt werden, ohne MACsec zu verändern [15]. Vom MSK werden alle anderen Keys abgeleitet, sodass dieser unbedingt geheim sein muss. Der Einsatz eines QKD-Keys als CAK ist somit entweder direkt als PSK oder auch im EAP als MSK denkbar, ohne das MKA-Protokoll abzuändern. Bei einer hierarchischen Schlüsselgenerierung ist es besonders wichtig, den zentralen Schlüssel auch in Zukunft quantensicher zu machen.

Doch es gibt noch weitere Einsatzgebiete mit Abänderung im MKA-Protokoll für QKD-Keys. Ein QKD-Key für **SAK** kann unter Abänderung des MKA und unter hohen QKD-Schlüsselraten ebenfalls eingesetzt werden. Einen Vorschlag für die Erweiterung des Protokolls liefert [15]. Unabhängig von MKA wird für MACsec eine Integration eines QKD-Keys für SAKs vorgeschlagen, die in jeder Sitzung neu generiert werden. Dies bringt den Vorteil mit sich, dass sonst abgeleitete SAKs anderer Sitzungen hier nicht bei Offenlegung des CAKs kompromittiert werden können. Hierfür wird eine hybride Schlüsselgeneration mit *Diffie Hellman* (DH) vorgeschlagen.

MACsec mit ETSI 014 bei Juniper

Bei Juniper Switchen und Routern ist sowohl ein statischer, als auch ein nicht-händisch generierter dynamischer CAK definierbar [16]. Zunächst muss eine CA mit einem PSK (CKN & CAK) definiert werden. Darin können die *Key Server-Priority* und das *Transmit-Interval* festgelegt werden. Für die erstellte CA müssen abschließend noch die Interfaces zur Konnektivität definiert werden. Der dynamische Modus wird ähnlich konfiguriert, bietet aber nicht die Optionen der CAs im statischen Modus. Für die Authentifizierung im dynamischen Modus mit Zertifikaten wird 802.1X verwendet.

Bei Juniper [17] wurden in der herstellerneutralen Testumgebung zwei QKD-Geräte (IDQ Cerberis XG 4. Gen) eingesetzt, sowie zwei Enkryptoren (Juniper SRX 380) mit MACsec-Encryption. Zusätzlich kam ein Abhör-Simulator von IDQ in einem Netzwerkaufbau in Nürnberg von der Deutschen Telekom zum Einsatz.

Dabei sind im praktischen Einsatz besonders drei Punkte aufgefallen, die es noch zu verbessern gilt:

1) Zuerst waren die Zeitbedingungen für die Key Requests & Delivery schwierig auf dem klassischen und dem Quantenkanal handzuhaben, da es keine Anzeige der zeitlichen Verfügbarkeit der Schlüssel in ETSI GS QKD 014 gibt.

Wie bei den übrigen Protokollen auch, ist die Lebensdauer der Sitzungsschlüssel, die mit einem Rekeying-Prozess einhergeht und mit einer Policy geregelt wird, essentiell für die Sicherheit. Die Frage nach einer durchschnittlichen Verwendungszeit ist nicht pauschal zu beantworten. Diese ist nicht nur vom jeweiligen Schlüssel (Kurz- oder Langzeit, Funktion) im jeweiligen Protokoll abhängig, sondern auch von der jeweiligen Anwendung: anfallende Datenrate, verfügbare Schlüsselrate, Sensibilität der übertragenen Daten, sowie individuelle Sicherheitsanforderungen, welche in den Rekeying-Intervallen festgelegt sind. Beim häufig eingesetzten symmetrischen Verfahren AES lassen sich unter MACsec etwa $2^{74,6}$ Bytes Daten mit einem Schlüssel mit XPN (64 Bit) verschlüsseln [15].

Die Zeitintervalle der Key Rotation Intervalle sind beliebig wählbar, variieren meist zwischen einigen Sekunden bis über einem Tag. Bei Juniper [17] wird beispielsweise alle 60 Sekunden ein Key Request zu den Key Management Entities (KMEs) geschickt und darauffolgend wird ein neuer CAK konfiguriert. Allgemein muss gelten, dass die Frequenz der Bereitstellung der QKD-Schlüssel größer gleich der aktiven Schlüssellebensdauer sein muss. Dafür muss ein passender Schwellwert ermittelt werden, um ein (zu schnelles) Leerlaufen der Schlüsselvorräte zu verhindern. Das Key Update Intervall bzw. der Key Request muss immer im Vergleich zur aktiven Schlüssellebensdauer (AKL) stehen, um auch Fehlerszenarien bei der QKD-Schlüsselbreitstellung überbrücken zu können. Die zwei möglichen Modi sind:

- **Modus 1: $f_{QKD} >> f_{AKL}$** bringt den Vorteil mit sich, dass ein paar gescheiterte Versuche der QKD-Schlüssel-bereitstellung abgepuffert werden können, bevor es zu einem Sicherheitsproblem, verursacht durch das Fehlen von Schlüsseln, kommt. Üblicherweise werden hier zwei Schlüssel gleichzeitig existieren: der derzeitig aktive Schlüssel und der zukünftige, bereits erhaltene Schlüssel.
- **Modus 2: $f_{QKD} \sim f_{AKL}$** bringt durch einen gescheiterten Key Request zwangsläufig ein Sicherheitsrisiko mit sich, was im Allgemeinen zu einem Kommunikationsstillstand führt, sollte es keinen Buffer mit ausreichend Schlüsseln geben. Zudem kommt es darauf an, für welchen Schlüssel – den CAK (quasi-statisch) oder den SAK (aus CAK erzeugt; häufig geupdatet) – ein sicherer QKD-Schlüssel eingesetzt werden soll. Während für den SAK, der durch die hohe Updaterate das Risiko eines Schlüsselmangels auf Dauer birgt, die Nutzung eines QKD-Schlüssels sich eher als schwierig gestaltet, wäre eher die Nutzung von QKD-

Schlüsseln für CAK sinnvoll, da SAKs als sicher angesehen werden können, solange der CAK geheim bleibt.

2) Als Zweites war die mangelnde Handhabung von Fehlerszenarien bei der Erzeugung von QKD-Schlüsseln zu beklagen.

3) Zuletzt ist die Schwierigkeit beim praktischen Einsatz von MACsec bei Juniper aufgefallen, allgemein zusätzliche Identifier auszutauschen. Eine Möglichkeit hierzu sollte in existierende Standards hinzugefügt werden, um z. B. einen Key-ID oder eine SAE-ID (Simultaneous Authentication of Equals ID) hinzufügen zu können. Für einen praktikableren Umgang werden zwei Identifier zusätzlich zu ETSI GS QKD 014 zur Implementierung der Kommunikation zwischen den MACsec Teilnehmern empfohlen einzusetzen: Zum einen die Key-ID und zum anderen die SAE-ID zur eindeutigen Identifikation der Teilnehmer. Jedoch muss dabei berücksichtigt werden, dass die derzeitige MACsec Control Plane nicht über die Möglichkeit verfügt, diese Identifier selbstständig abzurufen, sodass diese über das *Link Layer Discovery Protocol* (LLDP) ausgetauscht werden.

Proprietäre QKD-Protokolle zur Erweiterung von MACsec

Während MACsec selbst keine Protokolle bereitstellt, ist das MACsec Key Agreement Protocol (MKA) [14] dafür zuständig, dass Protokolle sowohl für die Authentifizierung als auch für den Schlüsselaustausch bereitstehen und die Schlüssel zwischen den Peers ausgetauscht werden. Das MKA-Protokoll ist allerdings dafür ausgelegt, interne statische PSKs meist einzeln manuell anzuwenden und davon die anderen Schlüssel selbst abzuleiten. Wenn ein Rekeying stattfinden soll, so müssen bei vielen Implementierungen alle PSKs bereits zum Definitionszeitpunkt komplett vorliegen mit definierter Lebenszeit, sodass ein automatischer Wechsel erfolgen kann. Dieser Mechanismus ist somit manuell nur schwer dynamisch umzusetzen und ist mit den derzeitigen Schlüsselraten, die von QKD-Geräten bereitgestellt werden (meist > 1.5 kbit/s) kaum praktikabel. Denn für den Einsatz von QKD-Keys sind dagegen PSKs *Out-of band* nötig. Dies bedeutet, dass ein synchronisierter, authentifizierter Schlüsselkanal als externe Quelle dauerhaft für zukünftige Schlüssel definiert werden muss. Die ETSI-Standards 004/014 sorgen für den nötigen sicheren Austausch der QKD-Schlüssel mit Enkryptoren. Somit stehen MKA QKD-PSKs dynamisch zur Verfügung. Die bisherigen ETSI-Standards müssen trotzdem für die Verwendung in der Praxis mit MACsec, optimiert werden, z.B. wie bei Juniper [17] vorgestellt. Eine Integrierung von QKD in MACsec ist bisher nicht standardisiert. MACsec wird zunehmend in den kommerziellen Switches/ Routern/ Firewalls mit Quantenschlüsseln implementiert. Schlüsselströme, die von QKD-Geräten bereitgestellt werden, sind jedoch nicht zertifiziert synchron abrufbar und stehen nicht immer für die direkte Verschlüsselung des Datenstroms passend zur Verfügung. Eine QKD-Erweiterung von MACsec wäre unbedingt nötig, um nicht auf proprietäre Protokolle einzelner Hersteller, wie z.B. Cisco SKIP zurückgreifen zu müssen, die wiederum nicht mit Geräten anderer Hersteller interoperabel sind. Zudem würde das die Integration in andere Protokolle erleichtern.

Cisco's SKIP

Das *Secure Key Integration Protocol* (SKIP) von Cisco ist speziell für den hardwareseitigen Einsatz von QKD-Keys in Cisco-Routern konzipiert worden. Es existiert auch ein erster Entwurf von der *Internet Engineering Task Force* (IETF) [18]. Dieser erschien erstmal im August 2024 und wird – wie jeder IETF-Draft – halbjährlich aktualisiert. Von der externen QKD-Hardware[2] wird ein Session-Key und eine zugehörige Key-ID bei Alice über eine TLS-Verbindung abgeholt. Zu Bob wird dann über den Kanal nur die zugehörige Key-ID übersendet, damit sich Bob auch den zugehörigen Schlüssel von seiner QKD-Instanz via TLS abholen kann. Wichtig ist, dass die QKD-Hardware dafür sorgt, dass die Schlüssel synchron sind. Allgemein definiert SKIP nur die vertikale Schnittstelle zwischen Key Provider (hier: QKD-Gerät) und Enkryptor auf beiden Seiten. Die horizontale Peer-to-Peer-Verbindung ist frei wählbar.

SKIP ist an sich layerunabhängig und kann für jedes QKD-geeignete Sicherheitsprotokoll implementiert werden. SKIP ist bei Cisco Switchen/Routern für Layer 2/3 mit ein paar Befehlen konfiguriert und bietet den Vorteil, dass die Schlüsselquelle als IP-Adresse fest konfiguriert werden kann. So können in einem bestehenden Netzwerk die QKD-Geräte als Schlüsselquelle definiert werden und die Schlüssel von dort automatisch bei Einstellung eines Rekeyings abgeholt werden. Außerdem kann eine entsprechende Policy (die z.B. auch das SAK Rekeying-Intervall angeben kann) mit einer MKA verknüpft werden. Diese kann dann wiederum über eine Keychain fest mit einem Port verknüpft werden, sodass eine vordefinierte Punkt-zu-Punkt Verbindung MACsec-verschlüsselt etabliert werden kann. Die Integration mit Cisco SKIP dürfte für eine Institution, die QKD in ihr Netzwerk einbinden möchte, somit aktuell die einfachste Möglichkeit darstellen.

Als Kommunikationsprotokoll stützt sich SKIP auf Hypertext Transfer Protocol (HTTP) über Transport Layer Security (TLS) 1.2/1.3, wobei bei letzterem der TLS_AES_256_GCM_SHA384-Algorithmus zum Einsatz kommt. Zunächst werden einmal die Möglichkeiten des Key Providers abgefragt. Folglich können Schlüssel abgefragt werden per ID (mit entsprechender Größenanforderung), jeweils von Alice und Bob, oder ein zufälliger Schlüssel. Als Antwort des Key Providers gibt es vordefinierte Statusmeldungen in Form von Fehlercodes und genaue Schemata, wie die Antwort des Key Providers aussehen muss.

SKIP erleichtert die einzelne manuelle *Out-of-band* Einspeisung von Pre-Shared Keys, besonders im Hinblick auf Key Management und Rekeying/Key Rotation Policies. Wie oben zu sehen, wird SKIP auch für MACsec eingesetzt und kann potentiell in jedem Protokoll eingesetzt werden, das mit Pre-Shared Keys arbeitet.

Dabei wird SKIP nicht in jedem Cisco Router/Switch unterstützt. Mit neuen Software-Releases wird es auf mehr Geräte ausgerollt. Allerdings müssen diese über eine kostenpflichtige MACsec-Lizenz verfügen, um die speziellen kryptografischen Eigenschaften nutzen zu können. Es ist ein sogenanntes *cryptopqc*-Feature zu aktivieren. In diesem lässt sich anschließend neben MACsec auch ein Quantum Key Distribution Profile einrichten unter Angabe einer Key Management Engine und eines Trustpoints, der alle Zertifikate enthält. Somit lässt sich ein gesicherter Kommunikationskanal einrichten, über den automatisch nach angegebener Expiration Time in der Policy neues Schlüsselmaterial synchron nachgeliefert wird. Somit ist ein automatischer Schlüsselstrom ermöglicht und ersetzt so, die manuelle Anforderung einzelner Schlüssel mit MKA:

Der Ablauf von SKIP ist unten Abbildung 6 dargestellt: Peer 1 fordert von seinem QKD-Gerät / Key Provider (KP) einen Schlüssel mit zugehöriger Key-ID an und schickt die Key-ID synchronisiert, aber

[2] Im Gegensatz zu Cisco's *Session Key Service* (SKS) bei dem nur mit einer lokalen Engine im Router software-basiert quanten-sichere Keys erzeugt werden, sind bei Cisco SKIP externe QKD-Gerätepaare auf beiden Seiten erforderlich.

unverschlüsselt seinem Peer 2. Es wird gefordert, dass aus der Key-ID sich keine relevanten Informationen über den Schlüssel ableiten lassen. Peer 2 fordert von seinem KP den zugehörigen Schlüssel an. Durch die Nutzung von Pre-shared Keys und Zertifikaten zur Authentifizierung von TLS, wissen KP und Enkryptor, wem sie die Schlüssel anvertrauen dürfen. Die Verbindung zwischen KP und Enkryptor ist mit TLS v1.2/3 abgesichert[3]. Dennoch ist dies nicht post-quanten sicher, auch Zertifikate können noch nicht post-quantum übertragen werden. Diese Schwachstelle muss unbedingt mit PQC oder physischer Absicherung in einem gemeinsamen Gehäuse überwunden werden. SKIP kann auch mit IKEv2 und PPK genutzt werden, wodurch sich dieser Link zwischen den Enkryptoren ebenfalls absichern lässt.

SKIP ist somit hilfreich bei sämtlichen Protokollen auf allen Schichten, in die ein Post-quantum Pre-Shared Key (PPK) integriert werden kann, denn es erleichtert das Zurverfügungstellen der PPKs von den QKD-Geräten um Vieles.

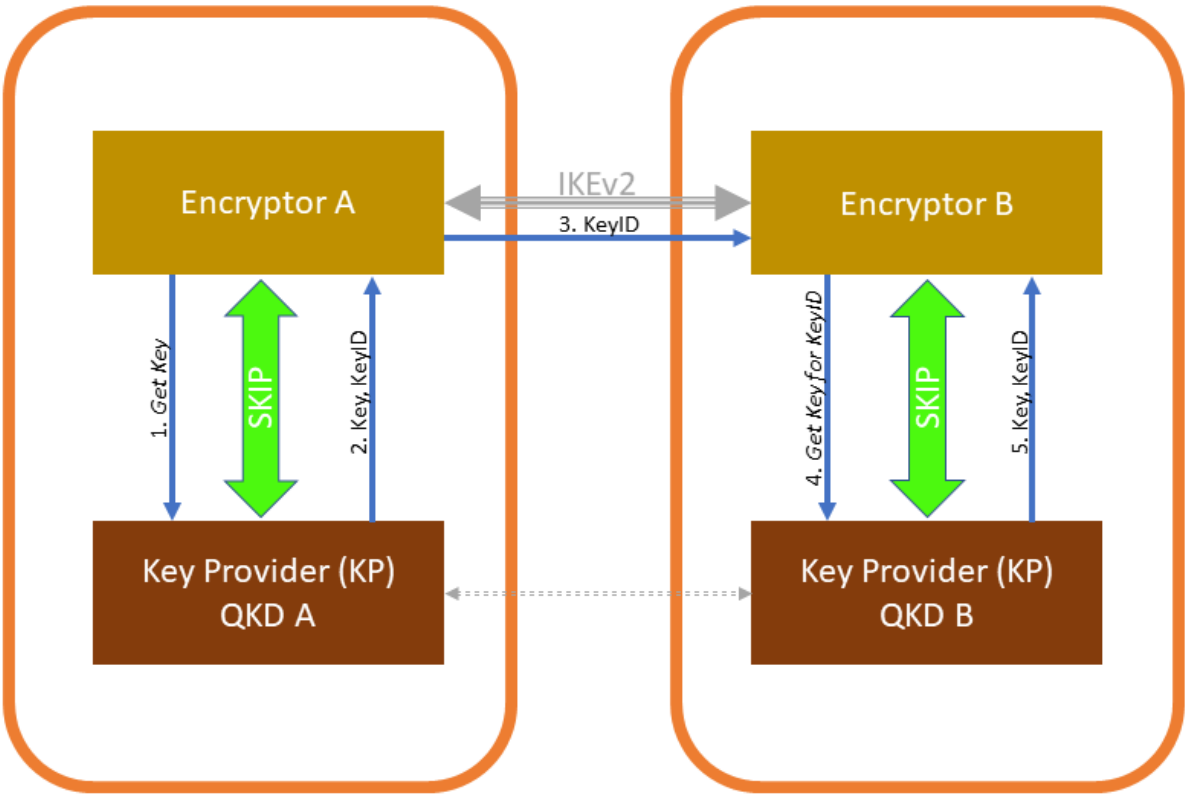


Abbildung 6: Zwei Trusted Nodes (orange) und die Kommunikation beim SKIP-Protokoll von Cisco: 1. Vom Enkryptor-A wird der Key bei KP-QKD-A angefragt. 2. Darauf sendet dieser KP: Key, KeyID zum Enkryptor-A. 3. Die KeyID wird über einen nicht-quantensicheren Kanal zu Enkryptor-B übertragen. 4. Mit dieser Key-ID fordet Enkryptor-B den symmetrischen Schlüssel von KP-QKD-B an. 5. Enkryptor-B erhält Key, KeyID vom KP-QKD-B zur Dekryption.

[3] Bei TLS 1.3 wurden zwar alle statischen RSA und DH Cipher Suites zur Verschlüsselung im Vergleich zu 1.2 entfernt, da diese asymmetrischen Algorithmen nicht quantensicher sind. Die Authentifizierung im Handshake von TLS 1.3 zur Generierung des Master Secrets beruht aber immer noch auf diesen asymmetrischen Verfahren. TLS 1.3 ist zwar standardmäßig nicht quantensicher, lässt sich aber quantensicher erweitern mit QKD/PQC PSKs (Kapitel „TLS 1.3“).

Nokia's ANYsec

Nokia's proprietäres ANYsec-Protokoll auf MACsec basierend, verschlüsselt Multiprotocol Label Switching (MPLS Layer 2.5) und bietet damit Enkryption von Layer 2 bis 3 mit geringer Latenz in jedem Netzwerk. Dabei werden allerdings nur die Nutzdaten mit GCM-AES 128/256 verschlüsselt, da die Header die Label enthalten, die bei MPLS die Informationen über die einmalige Routenberechnung beinhalten. So können die Label beim Switching durch das Netzwerk von jedem Router bearbeitet werden, während die Nutzdaten nur am Zielrouter, der auch ANYsec unterstützen muss, entschlüsselt. Auch eine Authentifizierung bleibt aus. Da ANYsec auf MACsec basiert, wird MKA als Kontrollebene mit vordefinierten PSKs nun mit leichten Abänderungen über IP eingesetzt.

Ein abstrakter Ablauf des Protokolls sieht wie folgt aus [19]:

1. Es werden ANYsec-spezifische CAs konfiguriert
2. Das MACsec Key Agreement (MKA) Key Server mit entsprechender Priorisierung erzeugt die SAKs lokal
3. Das MKA verschlüsselt die SAKs via CMAC-AES-128/256 mittels PSK
4. MKA sorgt für die anschließende Verteilung zwischen den Peers (Abänderung bei ANYsec zur Ermöglichung des IP-Transports)
5. ANYsec verschlüsselt via IEEE802.1AE Encryption Engine mit den SAKs die MPLS-Nutzdaten

Die Paketierung von ANYsec (Abbildung 7) zeigt nochmal den Unterschied in den unverschlüsselten und unauthentifizierten Labeln für den IP-Transport im Vergleich zu der von MACsec in Abbildung 4.

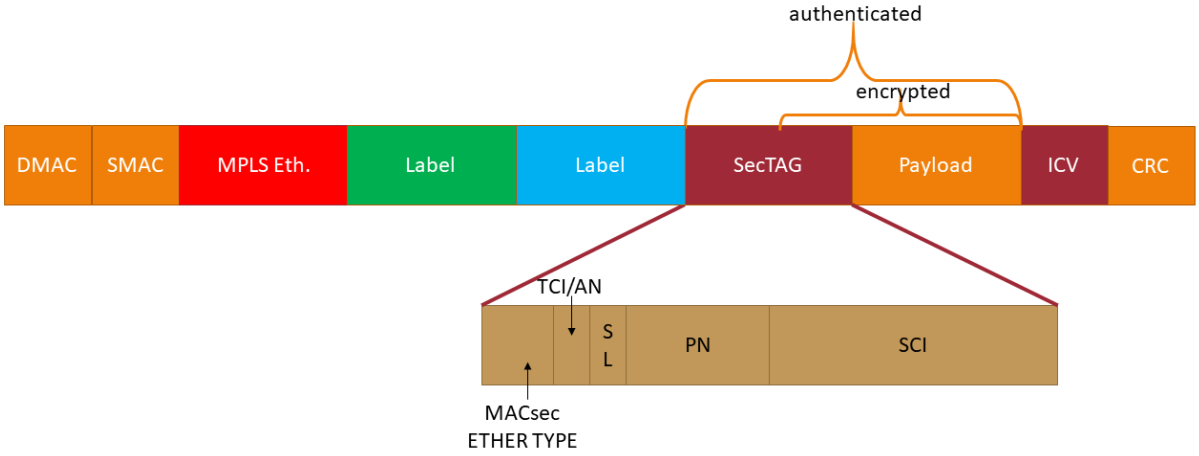


Abbildung 7: Die Paketierung von ANYsec: Header mit Destination Medium Access Control (DMAC)- und eine Source MAC-Adresse (SMAC), MPLS Ether Type, den unverschlüsselten Labeln zur Zuordnung der Frames, dem SecTAG mit MACsec Ether Type, der TAG Control Information (TCI), der Association Number (AN), der Shorth Length (SL) und der Packet Number (PN), optionaler Secure Channel Identifier (SCI), Integrity Check Value (ICV), zyklischen Redundanzcheck (CRC).

Somit lässt sich ANYsec bezüglich der Möglichkeiten zur Integrierung von QKD-Schlüsseln ähnlich bewerten. Die SAKs, die durch Zufallszahlen generiert werden und mit CMAC-AES-128/256 unter dem PSK verschlüsselt werden, sollten idealerweise auch durch QKD ersetzbar sein. Zudem ist ANYsec durch die große Ähnlichkeit zu MACsec flexibel damit zusammen einsetzbar.

Schicht 3

IPsec mit IKEv2

IPsec agiert auf dem Network Layer und soll durch Verschlüsselung Sicherheit gegenüber den IP-Protokollen bieten. Ein Vorteil bei der Verschlüsselung auf der Vermittlungsschicht besteht darin, dass darüberliegende Layer (wie Transportschicht oder Anwendungen) auf ein gemeinsames Key Management Netzwerk zugreifen können [20]. Dabei werden sowohl die Informationen über das Paket, als auch die im Paket enthaltenen Daten abgesichert. Daneben werden auch Zugangskontrolle, Datenintegrität, Authentifizierung des Datenursprungs, Vertraulichkeit der Daten und Anti-Replay Schutz gewährleistet. Dazu gibt es zwei Sicherheitsprotokolle: den *Authentication Header* (AH) mit AH-Header und die *Encapsulating Security Payload* (ESP) mit ESP-Header & -Trailer. Bei AH wird Authentifizierung und Integrität ermöglicht, aber keine Vertraulichkeit unterstützt, während ESP Authentifizierung, Integrität und/oder zusätzlich Vertraulichkeit durch Verschlüsselung ermöglicht. Deshalb wird ESP häufiger in der Praxis eingesetzt, sowie wegen weiteren operativen Vorteilen. Beide Protokolle unterstützen Transport und Tunnel Mode, je nachdem ob die Sicherheit auf die IP-Datagramme angewendet wird, oder auf die höheren Schichten. Die *Security Associations* (SAs) dienen zur Aushandlung von Sicherheitsparametern, u.a. Verschlüsselungsalgorithmus und verwendeten Schlüsseln und werden in der Security Association Database (SAD) mit einer Lifetime in Sekunden oder Kilobytes gespeichert, während die anzuwendenden Policies in einer Security Policy Database (SPD) hinterlegt sind. Soll QKD als neuer Algorithmus integriert werden, so muss dies in der SA geschehen.

Der *Internet Key Exchange* (IKE) wird bei IPsec für die automatische Erzeugung und das Management von IPsec SAs eingesetzt und basiert auf dem *Internet Security Association and Key Management Protocol* (ISAKMP). Dabei werden analog IKE SAs zur sicheren Verhandlung der IPsec SAs erzeugt. Beim Handshake muss sich folglich auf die IKE SA generell mit allen Bestandteilen des kryptographischen Algorithmus geeinigt werden: den Verschlüsselungsalgorithmus selbst, die Authentifizierungscodes und die Pseudo-Random-Funktionen zur Schlüsselgenerierung [21]. Zusätzlich werden mit IKE bei IPsec die Verbindungseinstellungen des Kommunikationskanals und die Authentifizierung zwischen den Teilnehmern verhandelt. Beim nicht abwärts-kompatiblen IKEv2 (RFC 7296 [22]) gibt es ebenfalls grob zwei Phasen (1. IKE SA- , 2. IPsec SA-Verhandlung). Mit der IKE SA werden also die anschließenden IPsec SA-Verhandlungen abgesichert, bevor in einer zweiten Phase basierend darauf die IPsec SAs (~Child SAs bei IKEv2 genannt) aufgebaut werden können, die anschließend die Nutzdaten absichern sollen. Laut Bundesamt für Sicherheit in der Informationstechnik soll die Lebenszeit der IKE-SA maximal 24 h und die der IPsec-SA maximal 4 h betragen [23]. Läuft eine IPsec SA ab, so wird dieselbe IKE SA wieder zur Generierung einer neuen IPsec SA verwendet. Für weitere Details empfiehlt sich [20]. Für diese Phasen gibt es konkret folgende Exchange-Messages:

- **IKE_SA_INIT:**
  Dieser Austausch dient zur Aushandlung der kryptographischen Parameter (z. B. Verschlüsselungs- und Hash-Algorithmen) und zum Aufbau eines gemeinsamen DH-Secrets. Zudem werden Nonces (Wort, Buchstaben- oder Zahlenfolge aur einmaligen Verwendung) ausgetauscht, dadurch wird garantiert, dass jede Sitzung eigene Schlüssel erhält, selbst wenn zwei Verbindungen mit identischen Algorithmen und denselben Schlüsseln gestartet werden würden. In diesem Schritt wird die Basis für einen sicheren Kanal gelegt; Authentifizierung findet im nächsten Schritt statt.
- **IKE_AUTH:**

Hier authentifizieren sich die beiden Kommunikationspartner gegenseitig (z. B. mit Zertifikaten oder Pre-Shared Keys). Gleichzeitig wird die erste IPsec SA erstellt, die für den eigentlichen Datenverkehr genutzt wird. Ab diesem Punkt existiert ein sicherer, verschlüsselter Kanal zwischen den Endpunkten.

- **CREATE_CHILD_SA:**
  Mit diesem Austausch können zusätzliche Child-SAs aufgebaut oder bestehende SAs neu verhandelt werden (z. B. für neue Schlüssel durch Rekeying). Dadurch bleibt die Kommunikation auch über längere Zeit sicher, ohne die IKE_SA komplett neu aufzubauen. Dabei ist zu beachten, dass im Perfect Forward Security Mode auch die IKE_SAs neu generiert werden müssen. Das passiert in der Regel allerdings nicht häufiger als bei den IPsec SAs. Die Struktur ist ähnlich wie bei IKE_SA_INIT, aber innerhalb des bestehenden sicheren Kanals.
- **INFORMATIONAL:**
  Dieser Nachrichtenaustausch wird nach der Schlüsselaushandlung genutzt und ist somit verschlüsselt. In diesem werden Statusinformationen, Fehler, Benachrichtigungen oder Löschbefehle (z. B. für alte SAs) übertragen. Es findet hier keine Aushandlung von neuen Schlüsseln statt. Damit dient INFORMATIONAL hauptsächlich der Verwaltung und Aufrechterhaltung der bestehenden IKE- und Child-SAs.

Dabei ist DH fest in IPsec (IKE_SA_INIT) als initiale Verschlüsselung vor der Authentifizierung und bei der Schlüsselgenerierung innerhalb der SAs zur Verschlüsselung des Nachrichtenverkehrs verankert (CREATE_CHILD_SA). DH sollte wegen seiner mangelnden Sicherheit gegenüber Quantencomputern und erhöhter Rechenkomplexität durch die Integration von extern bereitgestellten QKD-Keys mit Rekeying über ausreichend schnelle Schnittstellen ersetzt werden [24]. Dies stellt aber eine tiefgreifende Veränderung dar. Außerdem sind in der Praxis die Key Management Systeme bei großen Netzwerken noch zu langsam, sodass die QKD-Keys nicht schnell genug zur Verfügung stehen. Dies lässt sich z.B. mit dem Rapid Rekeying Protocol [25] für IPSec ohne IKE verbessern, welches für eine schnelle Schlüsselsynchronisation mit Puffern eingesetzt wird. Generell ist es hilfreich, möglichst viele SAs definiert zu haben, damit beim Rekeying schnell Verbindungen aufgebaut werden können.

Integration von QKD in IPsec

Die meisten höherschichtigen Sicherheitsprotokolle bestehen aus asymmetrischen (Authentifizierung) und symmetrischen (Verschlüsselungs) Methoden (~ hybride Kryptosysteme nicht zu verwechseln mit dem Hybriden Schlüsselaustausch aus Kapitel Hybridisierung mit QKD). Dabei kann bei der Authentifizierung meist ein unsicherer asymmetrischer Authentifizierungsmechanismus durch einen symmetrischen, quantenbasierten PSK ersetzt werden.

Zum Einsatz von Quantenschlüsseln bei IPsec gibt es grundsätzlich zwei Möglichkeiten: Die zu Bevorzugende ist, dass das QKD Service Interface (~ SDN/KMS) die Schlüssel der Anwendung zur Verfügung stellt. Die Anwendung kümmert sich um die folgende Verschlüsselung, Authentifizierung, etc.. Es besteht noch die Möglichkeit, dass das QKD Service Interface die Schlüssel behält und selbst die von der Anwendung angeforderte Nachricht verschlüsselt und entsprechend verarbeitet. Man sollte also den IKE dahin abändern, keine eigene Schlüsselaustauschfunktion zu berechnen und den externen QKD-Key als Shared Secret zu verwenden, anstatt eine parallele Schlüsselaustauschfunktion zu definieren.

Es sollte zudem eine quantensichere Authentifizierungsmethode verwendet werden, wofür auch QKD-Schlüssel als PSKs eingesetzt werden können. QKD-Keys können zusätzlich zur Verschlüsselung des IKE SA Verkehrs eingesetzt werden, welche andernfalls von den Shared Secrets berechnet wird. Abbildung 8 zeigt dazu eine mögliche Integration von QKD-Keys in IPsec, in der die Verschlüsselung und die Authentifizierung durch diese ersetzt werden können.

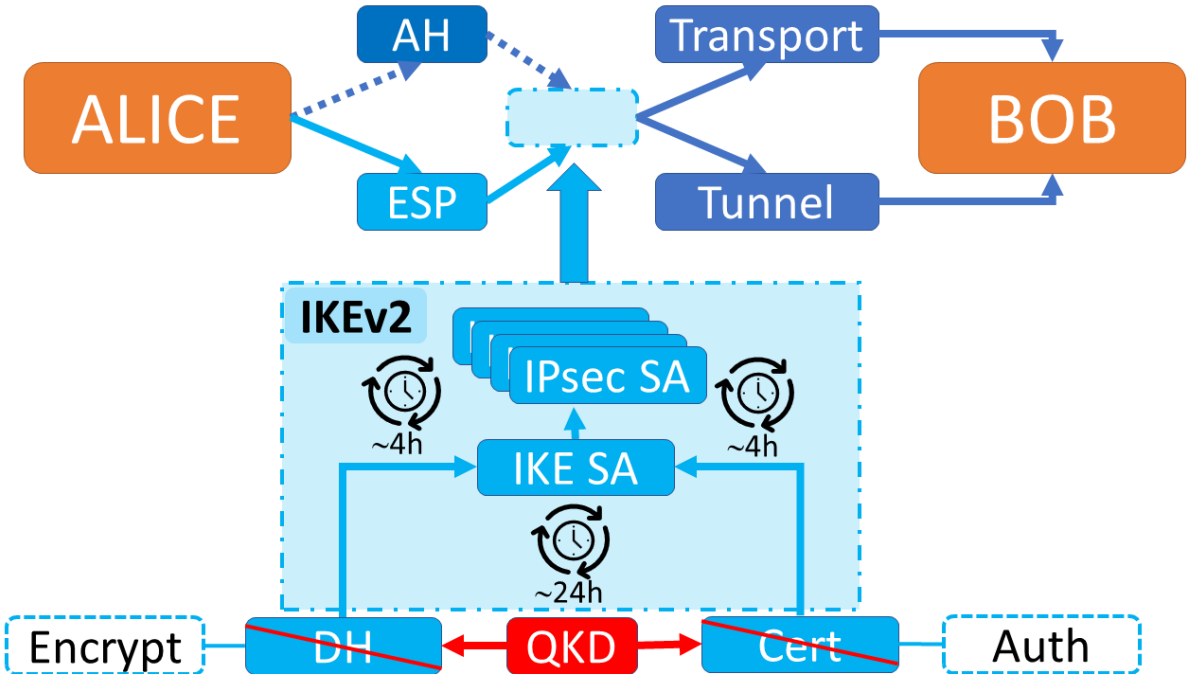


Abbildung 8: Mögliche Verschlüsselung von IPsec mit QKD, dabei wird nur ESP betrachtet. QKD Keys einmal als Ersatz für DH zur Verschlüsselung und einmal als Ersatz für die Authentifizierung (PSK). Dabei muss das jeweilige Rekeying berücksichtigt werden, da IKE SA (~24h Key) und IPSEC SA (~4h Key) unterschiedliche Schlüssel-Lebensdauern haben [23].

Auch die Nutzung zur Einmalverschlüsselung (OTP) wäre unter Berücksichtigung des erhöhten Schlüsselbedarfs, der entsprechend gleich der Wortlänge ist, als zusätzliche Sicherheitsverstärkung möglich. Denn die Anforderungen der Verschlüsselungstechnik One-Time-Pad (OTP) sind durch QKD erfüllt: Die Schlüssellänge muss mindestens gleich der Länge des Klartextes sein, sodass der Klartext mit einem zufälligen Schlüssel (dem Pad) derselben Länge (oder größer) durch eine bitweise XOR-Operation verschlüsselt wird. Die Zufälligkeit ist durch die Verwendung von QKD-Schlüsseln (BB84-Protokoll) ebenso erfüllt, wie dass der Schlüssel nur einmal verwendet werden darf und danach aus dem Speicher des Gerätes gelöscht werden muss.

Es müssen für die Unterstützung von QKD-Schlüsseln lediglich die Attribute im oben genannten IKE Handshake abgeändert werden. Zusätzlich ist sicherzustellen, dass die Quantenressourcen auch wirklich zur Verfügung stehen. Bleibt eine Antwort in der vorgegebenen Zeit aus, ist im IKE Protokoll ein alternatives Vorgehen für den Fall, in dem die Kommunikation nicht mehr aufgeschoben werden kann, implementiert: die Benutzung von pseudo-zufälligen Schlüsseln, welche immer im IKE auch für Authentifizierungszwecke benötigt werden. Allerdings muss bei der Verwendung von Einmalverschlüsselung die Anfrage an das IKE angepasst werden, da der Schlüsselbedarf unbekannt hoch ist. In einem solchen Fall, kann es auch zum Speicherplatzproblem der Schlüssel kommen, sodass nicht genügend Schlüsselmaterial bereitgestellt werden kann. [21].

Ein früher Vorschlag einer Erweiterung vom IPsec für QKD stellt [26] dar. Hier wird ISAKMP für den Gebrauch mit QKD erweitert, indem eine zusätzliche Phase am Anfang im Vergleich zu IKE eingeführt wird, in der symmetrische Schlüssel generiert werden. Die Erweiterung RFC 8784 für IKEv2 ermöglicht die zusätzliche Verwendung eines Pre-Shared Keys, der zum Beispiel statisch als PSK von QKD kommen

kann, nicht jedoch dynamisches Rekeying vor Ablauf der SAs unterstützt (möglich in [27]; Weiteres im Kapitel „Hybrider Schlüsselaustausch“. Dabei ist aber zu beachten, dass im Gegensatz zur quantensicheren Authentifizierung und IPsec-Verbindung, die initiale IKE-Verbindung zur Aushandlung der Sicherheitsparameter der SAs nicht ohne Rekeying quantensicher ist [1]. Der RFC 9370 bietet die Möglichkeit mehrere post-quantum Schlüsselaustauschfunktionen bei IKEv2 parallel zu definieren, sodass die Verschlüsselung mindestens so stark ist, wie jedes Verfahren individuell. Dennoch ergibt sich ein Problem mit der Schlüssellänge, was auch durch das Ermöglichen von Fragmentierung auf IKE Ebene durch RFC 7383 nicht völlig überwunden werden kann. Es gibt des Weiteren einen abgelaufenen Entwurf, der sich mit der Abänderung von IPsec (IKEv2) zur Integration von QKD beschäftigt [28]. Hier wurde eine Festlegung einer Key-ID und eines Fallback-Mechanismus in der SA-Payload vorgeschlagen.

Ciscos SKIP Protokoll kann auch mit IKEv2 genutzt werden. Gerade die Absicherung der Verbindung zwischen Enkryptoren und QKD-Geräten müsste SKIP-seitig um IKEv2 ergänzt werden. Idealerweise sollten bereits dazu QKD-Keys eingesetzt werden. IKEv2 in QKD-Testbeds wird in der Praxis beispielsweise beim MUQuaNet in München neben der Kerberos-Authentifizierung (siehe auch Kerberos und QKD) eingesetzt [29].

Schicht 4 & 5

TLS 1.3

Das *Transport Layer Security* (TLS) Protokoll (RFC 8446) [30] wird meist für die Verschlüsselung eines Tunnels zwischen Server und Browser (z.B. bei HTTPS, SMTP und SSH) benutzt, um Authentifizierung, Vertraulichkeit und Integrität zu erreichen. In einem ersten Schritt erfolgt ein Handshake, bei dem sich zwei Teilnehmer sicher auf eine gemeinsame Verschlüsselung und einen Authentifizierungscode einigen, sowie einer Key Agreement-Methode, um sich zunächst auf das sog. Pre-Master Secret festzulegen. Abbildung 9 zeigt eine Integration von QKD in TLS 1.3, das aus zwei Hauptkomponenten besteht, den TLS-Handshake und das TLS-Record Protocol. Zusätzlich werden die Pakete mit Sequenznummern versehen, um Umordnen/Replaying zu vermeiden. Im Handshake können verschiedene Algorithmen vereinbart werden und es können sowohl Public Keys als auch Pre-Shared Secrets ausgemacht werden. Das Record im zweiten Schritt kümmert sich um die Datenfragmentierung, -komprimierung & -verschlüsselung der eigentlichen Anwendungsdaten mit dem vorher vereinbarten Master Secret mit dem wiederum weitere Secret Keys generiert werden. Die Integration quantensicherer, symmetrischer QKD-Protokolle in TLS wird dadurch erleichtert, dass TLS nicht auf spezielle kryptographische Algorithmen festgelegt ist und außerdem bereits PSKs unterstützt werden (RFC 4279, RFC 5487). Dennoch haben nicht alle TLS-Implementierungen PSKs implementiert, aufgrund von mangelnden sicheren Austauschverfahren und Standardisierungen. [31] zeigt eine Möglichkeit auf, wie mittels „PSK Identity Hint“ im modifizierten ETSI 014 dennoch Client und Server sich einfach auf die Verwendung von PSKs einigen können.

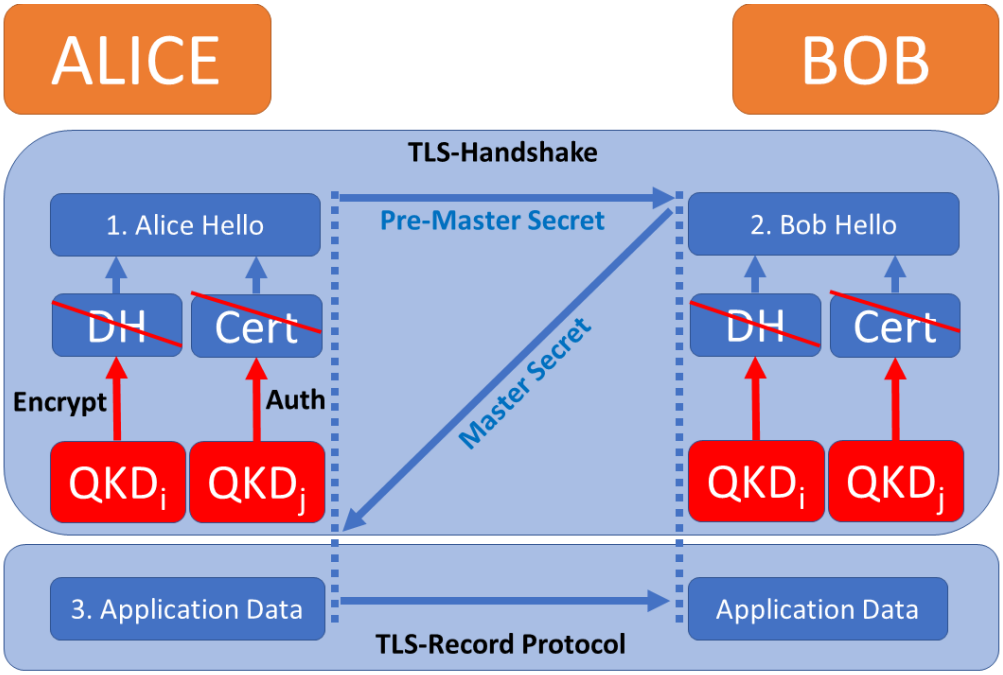


Abbildung 9: Mögliche Integration von QKD in TLS 1.3. Alice sendet „Hello“ Nachricht an Bob, in diesem Schritt wird sich auf Verschlüsselung und Authentifizierung auf den jeweiligen Seiten geeinigt. Dabei werden QKD Keys als Master Secret im TLS-Handshake einmal als Ersatz für DH zur Verschlüsselung und einmal als Ersatz für die Authentifizierung benutzt (PSK). Danach werden im TLS-Record Protocol die Anwendungsdaten mit den vorher vereinbarten Verschlüsselungen verschlüsselt.

Bei den meisten Sicherheitsprotokollen kommen hybride Kryptosysteme bestehend aus asymmetrischen (Authentifizierung) und symmetrischen (Verschlüsselung) Methoden zum Einsatz (nicht zu verwechseln mit dem Hybriden Schlüsselaustausch aus Kapitel „Hybridisierung mit QKD“). Während bei TLS ebenfalls asymmetrische und symmetrische Schlüssel verwendet werden, werden

beim TLS-Handshake asymmetrische Algorithmen für die Generierung des Master Secrets und für die anschließende Verschlüsselung symmetrische Algorithmen für die Session Keys eingesetzt. Daraus folgt, dass QKD-Keys sich bei TLS ohne tiefgreifende Protokolländerungen kaum einsetzen lassen. Für den Einsatz von QKD-Keys müsste die asymmetrische Authentifizierungsmethode komplett durch das PSK-Authentifizierungsverfahren ersetzt werden. Auch die Berechnung des Master Secrets könnte direkt durch einen QKD-Schlüssel ersetzt werden oder es könnten jeweils die generierten Zufallszahlen des Clients und Servers bzw. das Pre-Master Secret, von denen sich das Master Secret ableitet, mit QKD ersetzt werden.

OpenSSL wurde in einem Hackathon 2019 [32] dazu manipuliert, QKD-Schlüssel zu verwenden. Dazu wurden zwei Herangehensweisen vorgestellt: Mit sog. Engines können Erweiterungen von Dritten in OpenSSL integriert werden. Diese werden dynamisch in die Bibliothek geladen, sodass sich damit zwei Callback-Funktionen überschreiben lassen. Wie Abbildung 10 zeigt, lässt sich mit dem Engine Callback der Public-Key, der von DH ausgehandelt wird, durch einen Key-Handle mit dem QKD-Server ersetzen. In einem zweiten Engine Callback wird - während eigentlich der DH-Key generiert wird, stattdessen der QKD-Key von der ETSI-API erhalten. Dies beschreibt eine instabile Möglichkeit im Vergleich zu anderen im Rahmen der bestehenden Funktionen von OpenSSL, ohne das ursprüngliche Protokoll zu brechen. Alternativ besteht eine andere Herangehensweise in der QKD als neues Key-Exchange-Protokoll definiert wird. Auch hier wird eine Engine benutzt, um diesmal den QKD-Key von der ETSI-API entgegenzunehmen. Somit bleibt das Konzept der Engines eine gute Methode zur Integration von QKD in OpenSSL.

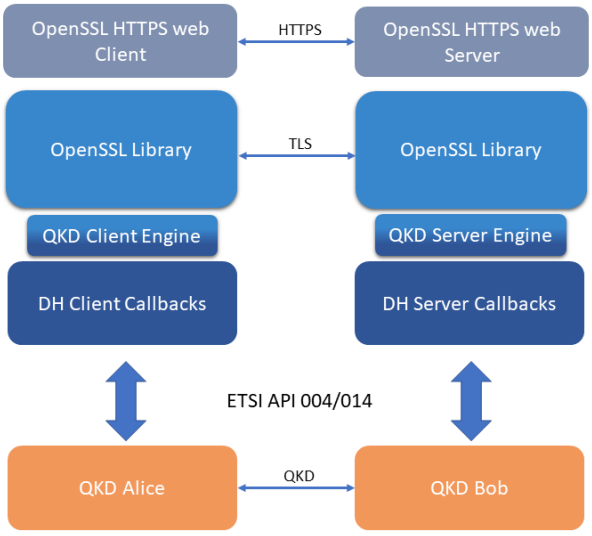


Abbildung 10: Beispiel für die Ausnutzung von dynamisch geladenen Engines für die Integration von QKD-Keys in OpenSSL [32]: QKD Client Engine als Bindeglied bei QKD Alice und QKD Server Engine als Bindeglied bei QKD Bob zwischen den DH Callbacks und der OpenSSL Libary. Die DH Client/Server Callbacks sind jeweils mit der ETSI-API an die QKD-Geräte angebunden, hierbei wird der Public Key durch den QKD-Key ersetzt. Dabei bleibt der Quellcode von OpenSSL unverändert, ledigilich die Schnittstelle wird in diesem Beispiel verändert bzw. gehackt.

In [21] wird QKD in das TLS-Protokoll integriert, indem der QKD-Key der Anwendung zur Verfügung gestellt wird und dieser dort weiter verarbeitet werden kann, wozu auch das Handshake abgeändert wird. Dabei ist dies bei TLS nicht so einfach, da es keine festgelegte Schlüsselaustauschfunktion gibt. Generell besteht auch bei TLS die Möglichkeit die QKD-Schlüssel als Master Secret, direkt für die TLS-Verschlüsselung, die Authentifizierungscodes oder die Einmalverschlüsselung mit einem interaktiv anpassbaren Stream Interface für QKD-Schlüssel einzusetzen. Bei TLS ist es besonders wichtig, die

Verfügbarkeit von QKD-Schlüsseln über den Quantenkanal abzufragen, um Strategien für Nichtverfügbarkeit zu erweitern, da in TLS keine Standardstrategie dafür definiert ist. Auch hier werden immer zusätzliche pseudo-zufällig erzeugte Schlüssel neben den verwendeten QKD-Schlüsseln u.a. für Authentifizierungszwecke benötigt. [21]

Gerade die Schnittstellen zum Erhalt der Schlüssel sind in der Praxis häufig der limitierende Faktor, wenn es um die Schnelligkeit der Integration von QKD-Schlüsseln im Vergleich zu klassischen Verfahren, die Bestandteil des Protokolls sind, geht [33]. [34] zeigt, wie man zum Zweck einer Videokonferenz, TLS 1.3 mit ETSI 014 QKD erweitern kann. Dennoch benötigt auch dieses Protokoll für das Handshake länger (trotz Public Key Encryption) als das klassische TLS-Protokoll, sodass es zu signifikanten Verzögerungen für die Anwendung kommen kann. Das schnellere QUIC-Protokoll basierend auf dem User Datagram Protocol (UDP) wird in [35] mit BB84 für einen schnellen TLS-Handshake kombiniert.

Zur Quantum-resistenten Transport Layer Security wird in einem niederländischen Testbed ein hybrider Ansatz dargestellt, um sowohl PQC als auch QKD in TLS 1.3 zu integrieren. Der Einsatz von QKD verzögert das Generieren und Verteilen des gemeinsamen Shared Secrets in TLS sehr, was eine unerwartete Beobachtung ist, weil nur 36 Bytes auszutauschen sind [36]. Ebenfalls ein hybrides Verfahren zu Integration in TLS 1.3 mit OpenSSL Providern stellt [37] dar. OpenSSL bietet zwei verschiedene Möglichkeiten zur Schlüsselbereitstellung, darunter auch das Key Encapsulation Mechanism (KEM) Interface, für Fälle wie QKD, in denen ein Teilnehmer den Schlüssel auch für die andere Partei bereitstellt. Dabei wird neben dem Server als Initiator, wie bei [36], auch ein Ansatz vorgestellt, bei der der Client als Initiator ermöglicht wird. Diese entwickelten Protokolle basieren ganz konkret auf ETSI 004/014 Schnittstellen, während jedoch die sequentiellen Sessions von ETSI 004 nicht mit der Server-Initiierung im Einklang stehen. Dieses Paper demonstriert, dass trotz tiefgreifender Veränderungen an beiden Protokollen, die an sich sehr unterschiedlich zu QKD funktionieren, trotzdem eine quantensichere Anpassung potentiell machbar ist.

Der Internet Draft Hybrid Key Exchange in TLS 1.3 [38] ermöglicht darüber hinaus eine Kombination von klassischen und PQC-Algorithmen über Identifier festzulegen. Öffentliche Schlüssel werden direkt im Handshake mitgeschickt, was diesen auch stark anwachsen lässt und den Overhead um einiges vergrößert.

Das klassische TLS-Protokoll wird außerdem im SKIP Protokoll von Cisco eingesetzt zur Absicherung der Kommunikation zwischen Enkryptor und QKD-Gerät (vgl. Cisco’s SKIP). Hier wäre ebenfalls eine Absicherung mit quantensicheren Schlüsseln nötig, um neue QKD-Keys sicher zu übermitteln.

Für die Lebensdauer der Sitzungsschlüssel bei AES-GCM lassen sich unter TLS 1.3 bis zu $2^{24.5}$ Datensätze (Records) auf einer Verbindung verschlüsseln, was etwa $2^{38.5}$ Bytes entspricht [15]. Für Anwendungen, die auf einen einzigen sicheren Schlüssel setzen, bedeutet das in den meisten Fällen, dass ein Schlüssel für mehrere Jahre ausreicht. Doch dafür ist QKD nicht ausgelegt, hier stehen so viele Schlüssel zur Verfügung, dass es für Rekeyings alle paar Minuten ausreicht. Im Gegensatz zur Verwendung von AES-Verschlüsselungen reicht die typische Schlüsselrate von kommerziellen QKD-Geräten von 1.5 kbit/s jedoch nicht für Einmalverschlüsselungen (OTP) aus, da es Schlüssel erfordert, die die gleiche Länge wie die zu übertragende Nachricht haben; aus diesem Grund ist die Übertragungsrate bestenfalls so hoch wie die Rate, mit der geheime Schlüssel erzeugt werden.

Es gibt jedoch einige Ansätze dazu, wie man dieses Problem beispielsweise optimieren kann: Zum einem ist es möglich einen linearen Optimierungsalgorithmus für die Verteilung der QKD-Keys zu definieren. Dabei werden anhand von unterschiedlichen Schlüsselraten an unterschiedlichen Knotenpunkten im Netz optimierte Pfade, zur optimalen Auslastung der Schlüsselraten, zwischen

diesen Punkten festgelegt [39]. Zum anderen können verlustfreie Komprimierungsalgorithmen für die Daten, die verschlüsselt werden sollen, benutzt werden. So kann zum Beispiel die Huffman-Kodierung verwendet werden, um Daten, die über den klassischen Kanal übertragen werden, zu komprimieren. Dies führt zu einem geringeren Overhead bei dieser Übertragung und zu einer Optimierung der benötigten Schlüsselraten mit QKD bei einer OTP-Verschlüsselung [40].

VPN Protokolle

Ein VPN (Virtual Private Network) ist ein Dienst, der eine sichere, verschlüsselte Verbindung zwischen einem Gerät und dem Internet herstellt und ist so ein Protokoll in Layer 3. Dieses kann wie ein „privater Tunnel“ in einem öffentlichen Netz beschrieben werden und ermöglicht Nutzern auf private Netzwerke zuzugreifen, als wären sie physisch miteinander verbunden. Eine gute VPN-Lösung sollte alle folgenden Punkte berücksichtigen [41]:

- Durch Verschlüsselung Schutz der Daten vor Abhörung
- Schutz vor Manipulationen der Pakete durch Verwendung von Hash-Funktionen, um die Integrität dieser zu gewährleisten.
- Schutz vor Man-in-the-Middle-Angriffen durch die Verwendung von Mechanismen zur Identitätsauthentifizierung.
- Schutz vor Wiederholungsangriffen (Replaying) durch die Verwendung von Sequenznummern bei der Übertragung geschützter Daten
- Definition der Mechanismen, wie Daten gekapselt und geschützt werden und wie geschützter Verkehr zwischen Geräten übertragen wird
- Festlegung, welcher Datenverkehr tatsächlich geschützt werden muss.

Es gibt mehrere Möglichkeiten von VPN-Protokollen, in dem QKD-Schlüssel benutzt werden können. Eine einfache Implementierung ist eine modifizierte Version der RFC 7296 IKEv2-Implementierung zu verwenden. Bei dieser wird der Einrichtungsprozess der Child Security Association (Child SA), also die Verwendung von QKD-Schlüsseln als Eingabe für den Schlüsselableitungsprozess erlaubt und die QKD-Schlüssel werden als neue Entropiequelle anstelle eines Diffie-Hellman-Schlüsselaustauschs, der - wie oben beschrieben - nicht quantensicher ist, verwendet [42]. Natürlich gibt es auch Möglichkeiten QKD in Software VPN-Anwendungen zu integrieren. Zwei Beispiele werden im Folgendem näher erläutert:

OpenVPN

OpenVPN ist ein VPN-Protokoll, das sichere Punkt-zu-Punkt-Verbindungen gewährleistet. Es wurde 2001 von James Yonan entwickelt und steht seitdem als Open-Source Software zur Verfügung. Es gilt bislang als einer der zuverlässigsten und flexibelsten VPN-Protokolle. Standardmäßig wird dieses Protokoll mit AES-256 (SSL/TLS) verschlüsselt und unterstützt sowohl UDP als auch TCP als Transportprotokolle. Durch seine jetzt mehr als 20 Jahre andauernde Zuverlässigkeit, gilt dieses als sehr sicher, etabliert und geprüft. Natürlich merkt man an der ein oder anderen Stelle, dass diese Software schon in die Jahre gekommen ist. Man hat im Vergleich zu anderen VPN-Protokollen wie zum Beispiel WireGuard einen deutlich komplexeren Implementierungsaufwand mit circa 400.000 Zeilen Code (WireGuard circa 4000 Zeilen) und ist so deutlich schwerer zu prüfen. Nichtsdestotrotz ist OpenVPN seit über 20 Jahren erprobt und findet breite Unterstützung vor allem wenn man maximale Kompatibilität für ein großes Unternehmensnetzwerk braucht.

Die Implementierung von QKD in OpenVPN läuft über die Standard-Schnittstelle ETSI 004, in dem die Schlüssel aus dem KMS Layer des Netzwerks extrahiert werden. Dieser schematische Aufbau ist in Abbildung 11 dargestellt.

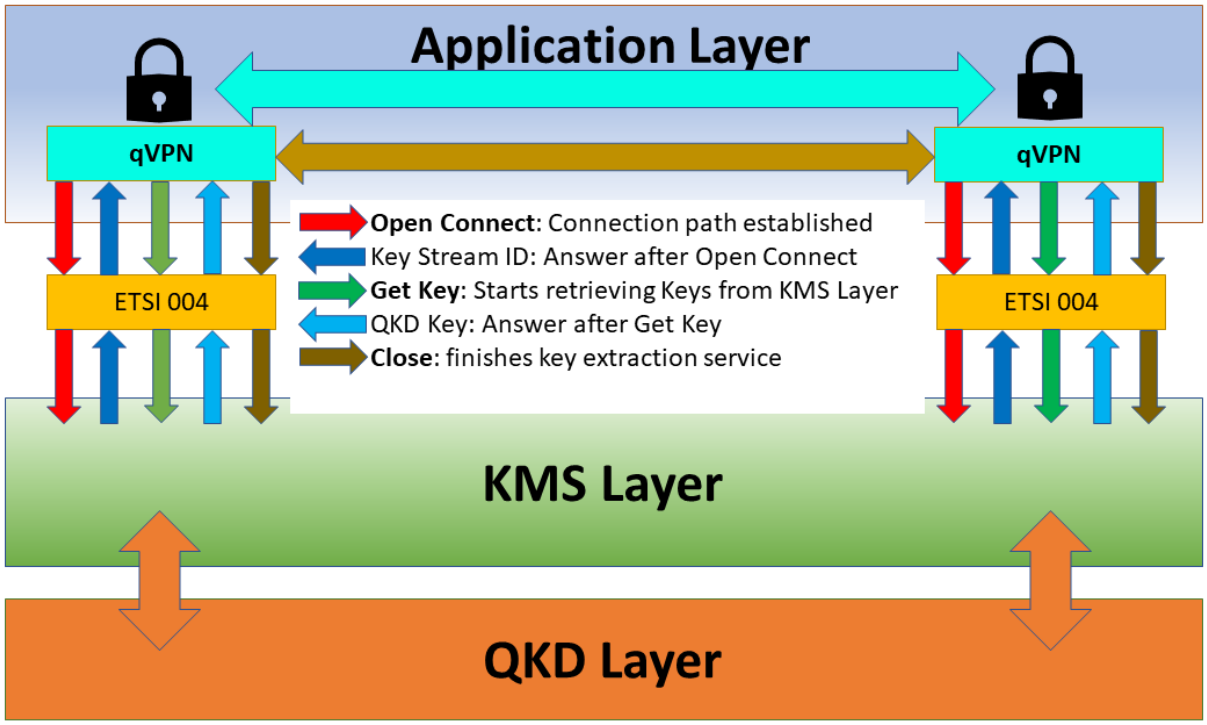


Abbildung 11: QKD-VPN: Struktur von VPN in Zusammenhang mit QKD In verschiedenen Schichten. Zuerst wird eine Verbindung über die Funktion „Open Connect" aufgebaut, dabei wird eine Keystream ID erzeugt, über die man die symmetrischen Schlüssel an beiden Endpunkten einer QKD-Verbindung über die ETSI 004 Schnittstelle im KMS-Layer abrufen kann. Über die „Get Key" Funktion können nun die einzelne Schlüssel, die im QKD-Layer erzeugt wurden, über die vorher generierte ID auf beiden Seiten abgefragt werden. [43]

Da dieses Protokoll in der Regel mit TLS verschlüsselt wird, kann die Methodik einer Integration von QKD-Keys in Kapitel Schicht 4 & 5 nachgelesen werden. OpenVPN verwendet also primär eine zertifikatsbasierte Authentifizierung, es gibt jedoch hierbei verschiedene Implementierungsmöglichkeiten für die Authentifizierung der jeweiligen Knotenpunkte mit QKD:

- Einmal besteht der klassische Fall mit dem Austausch von Server-Zertifikaten bzw. auch CAs, der in der Regel bei einem Verbindungsaufbau auch ohne QKD zum Einsatz kommt. Bei diesem muss der Client eine VPN_Hello Nachricht senden, die den QKD-Anwendungsidentifikator enthält, um die QKD-Verbindung herzustellen. Beim Eintreffen dieser Nachricht am Server ist es notwendig, dass beide Peers ihre OpenVPN-Konfigurationen mit der CA, dem Schlüssel, den Zertifikaten usw. vervollständigen. Sobald alle Konfigurationsparameter ausgefüllt sind, erstellt der Server eine QKD-Anwendungskennung, die er in der VPN_WELCOME-Nachricht an den Client übermittelt. Nun werden die beiden Endpunkte der QKD-Nodes verbunden, und die KMS-Schicht beginnt damit, die Schlüssel von den QKD-Modulen abzurufen. Um den anderen Endpunkt über die erfolgreiche Verbindung ihrer QKD-Anwendung zu benachrichtigen, sind beide Parteien dafür zuständig, eine Bestätigungsnachricht zu senden. Wenn die jeweiligen KMS-Schichten genügend Schlüssel abgerufen haben, fordern beide Anwendungen einen von QKD erhaltenen Schlüssel an, um einen sicheren VPN-Dienst zu konfigurieren. Dieser Dienst wird alle 2 Minuten mit den neuen, von QKD abgeleiteten Schlüsseln aktualisiert, um ein Rekeying zu ermöglichen [43].

- Eine weitere Möglichkeit für die jeweilige Authentifizierung ist die ohne einen Austausch von Zertifikaten. Anstelle von diesen erzeugt das Quantenverfahren zwei symmetrische Schlüssel für Alice und Bob, d.h. diese erhalten zwei identische Binärstrings. Für die Authentifizierung veröffentlicht Alice den ersten Teil dieses Binärstrings, den Bob mit seinen eigenen Messergebnissen vergleicht. Stimmen die beiden Binärzahlen überein, ist Bob sicher über die Identität seines Kommunikationspartners, und Alice gilt als authentifiziert. Bobs öffentlicher Schlüssel wird nach demselben Verfahren aus den zweiten Teil dieses Binärstrings gebildet, wodurch Bob vollständig authentifiziert wird. Tritt bei diesem Vergleich eine Abweichung auf, brechen beide Parteien das Protokoll ab und starten eine neue Sitzung. Mit diesem Schlüssel wird im späteren Verlauf das Master Secret und die restlichen Schlüsselhierachie abgeleitet. Das bedeutet, dass die Authentifizierung alleine von den Schlüsseln abhängt, die vom Quantenalgorithmus erzeugt wurden. Dieses Quantenschema ersetzt die Public-Key-Infrastruktur (PKI), um die Authentifizierung der beiden Parteien vorzunehmen und einen Geheimnisabgleich zu ermöglichen. Das vorgeschlagene Protokoll ist effizienter, da es auf Quanteneffekten basiert, um einen Quantenrahmen zu schaffen, der die Zertifikate überflüssig macht. Zusätzlich zur Quantenauthentifizierung erfolgt eine zweite Authentifizierungssitzung durch den Austausch der fertigen Nachrichten. Bei gelungener Quantenauthentifizierung haben die beiden Endpunkte dasselbe Geheimnis, das zur Berechnung dieser Nachricht verwendet wird [44].

WireGuard

WireGuard wurde im Jahr 2016 von Jason A. Donenfeld entwickelt [45]. Es ist im Vergleich zum etablierten OpenVPN Protokoll deutlich schlanker, einfacher zu realisieren und schneller in der Übertragung. Dieses Protokoll nutzt moderne Verschlüsselungsmethoden wie zum Beispiel ChaCha20 [46] oder Curve25519 [47], unterstützt aber leider nur UDP als Transport Protokoll. Darüber hinaus enthält es fortschrittliche Sicherheitsmerkmale wie Identity Hiding (IH) und Perfect Forward Security (PFS) durch bis zu vier Runden von Elliptic-Curve Diffie-Hellman (ECDH)-Operationen. Es wurde allgemein dafür entwickelt sicherer zu sein als traditionelle bzw. etablierte VPN Protokolle, dennoch stehen diese etablierten Protokolle in puncto Sicherheit bei der richtigen, aber dennoch aufwändigen, Implementierung WireGuard in nichts nach. Trotzdem dank seiner Effizienz mit geringen Latenzen und hoher Performance eignet sich dieses Protokoll besonders gut für eine schnelle Implementierung in z.B Mobil-Geräte oder eingebettete Systeme.

WireGuard sollte, um eine quantenresistente Sicherheit zu erzielen, die Verwendung anfälliger asymmetrischer Kryptographie-Algorithmen vermeiden, die derzeit eng mit dem WireGuard-Protokoll verbunden sind. Abbildung 12 zeigt eine solche Möglichkeit zur QKD-Integration, in der das Authentifizierungsverfahren auf Tunnel-Level anstatt auf asymmetrische Kryptographie Algorithmen für die gegenseitige Authentifizierung beruht. Dabei ist es für WireGuard-QKD notwendig, dass der Initiator und der Responder-Peer auf Tunnelebene einen PSK pflegen. Jeder Knoten erstellt einen zufälligen statischen sicheren Schlüssel, der für das Authentifizierungsgeheimnis während des Handshake-Prozesses verwendet wird. Zudem ist das Ziel alle ECDH Operationen im Handshake-Prozess zu ersetzen, da diese anfällig für Quanten Angriffe sind.

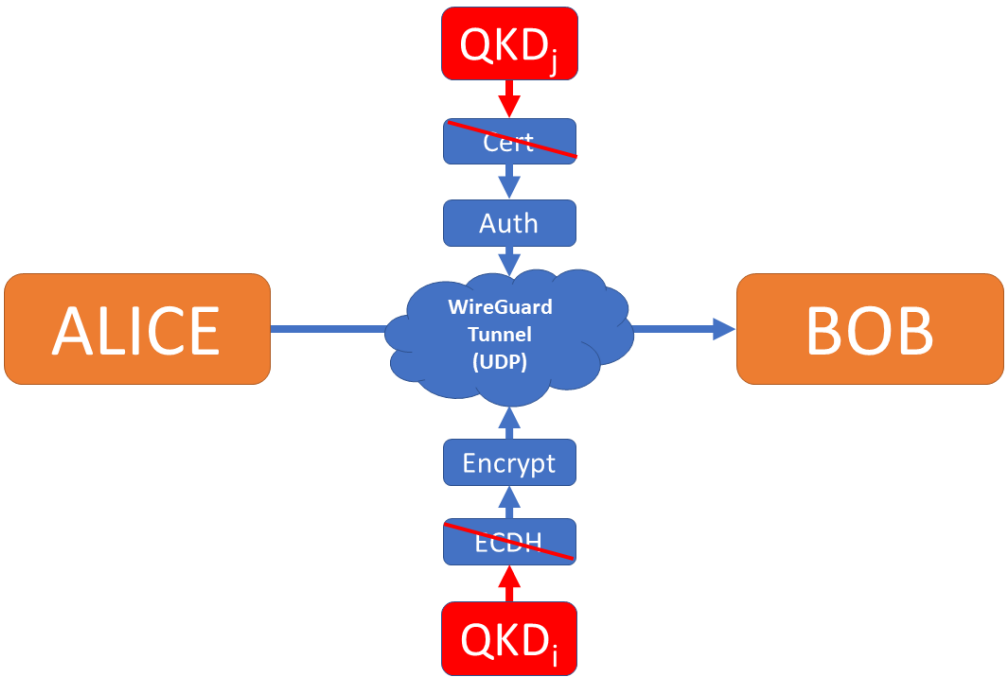


Abbildung 12: Mögliche Integration von QKD in das WireGuard Protokoll. Dabei werden QKD Keys einmal als Ersatz für ECDH zur Verschlüsselung und einmal als Ersatz für die Authentifizierung (PSK) benutzt.

Das ursprüngliche WireGuard-Protokoll implementiert ein konstantes Rekey-Schema, das alle etwa 90 Sekunden Handshakes durchführt. Allerdings resultiert daraus eine ineffiziente Verwendung der QKD-Schlüssel, da diese schneller erzeugt werden als verbraucht. Daher wird in [48] eine dynamische Anpassung der Rekey-Periode des Handshakes vorgenommen, damit die QKD-Schlüssel vollständig verwendet werden können und daraus resultierend die Wiederverwendung von Schlüsseln weiter zu minimieren, um eine One-Time-Password-Verschlüsselung zu ermöglichen.

Beispiele für Dienste für sichere Netzwerkverbindungen

EduVPN

EduVPN ist ein VPN-Dienst, der speziell für die Bedürfnisse von Bildungs- und Forschungseinrichtungen entwickelt wurde. Dieser wurde von dem niederländischen National Research and Education Network (NREN) SURF entwickelt. Er wird von der internationalen Organisation GÉANT, ein Zusammenschluss europäischer NRENs, betrieben, um Mitarbeitenden und Studierenden eine sichere und datenschutzfreundliche Verbindung zu ihrer Hochschule oder Institut zu ermöglichen – unabhängig davon, wo sie sich gerade befinden.

Ein großer Vorteil von EduVPN ist die einfache Nutzung mit gleichzeitig hohem Sicherheitsstandard. Sowohl OpenVPN als auch WireGuard werden von EduVPN unterstützt. WireGuard wird zunehmend von EduVPN bevorzugt, da es leichter zu warten ist, schneller arbeitet und ein geringeres Fehlerpotenzial aufweist. Trotzdem bleibt OpenVPN als bewährte und flexible Option in vielen Netzwerken im Einsatz, insbesondere dort, wo Kompatibilität zu älteren Systemen gefragt ist. EduVPN ist kostenlos, Opensource und wird dauerhaft von GÉANT weitergepflegt und bietet so eine sinnvolle billige alternative zu kommerziellen Varianten. Dieses entspricht damit auch der vom LRZ München und von der FAU Erlangen-Nürnberg ausgegebenen Handlungsempfehlung für die Bayerischen Hochschulen [49].

Eine Verschlüsselung mit QKD ist bei weitem noch nicht standardmäßig implementiert, da aber eduVPN sowohl OpenVPN als auch WireGuard benutzt, kann eine mögliche Integration von QKD in den jeweiligen Kapiteln „OpenVPN“und „WireGuard“ nachgelesen werden.

Eduroam

Eduroam wurde ursprünglich 2002 von der TERENA (Trans-European Research and Education Networking Association, heute Teil von GÉANT) entwickelt. Dieser ist ein weltweit verbreiteter WLAN-Roaming-Dienst für Forschungs- und Bildungseinrichtungen. Er basiert auf dem IEEE-802.1X-Standard in Kombination mit einem hierarchischen RADIUS-Infrastrukturmodell, das eine zentralisierte Authentifizierung über verschiedene Institutionen hinweg ermöglicht [50].

Nutzer authentifizieren sich mit den Zugangsdaten ihrer Heimateinrichtung. Die Zugangsanfrage wird über TLS-verschlüsselte RADIUS-Tunnel (meist Extensible Authentication Protocol (EAP) - Methoden wie Protected - EAP oder EAP-Tunnel-TLS) durch die zentralisierte Struktur bis zum Identity Provider (IdP) der jeweiligen Heimatorganisation geleitet. In der Regel werden Daten im WLAN mittels Wi-Fi Protected Access (WPA)2-Enterprise oder WPA3-Enterprise verschlüsselt übertragen [51].

Eine Integration von QKD in Eduroam ist in der Regel nur sehr schwer umsetzbar bzw. eigentlich nicht möglich. Eduroam ist mit WPA-Enterprise entworfen, das auf individueller Authentifizierung mit dynamischer Schlüsselvergabe basiert. So macht ein PSK-Konzept in diesem Modell keinen Sinn – es würde die zentralen Sicherheitsvorteile aufheben. Die Schlüsselaushandlung ist wie folgt [52]:

1. Authentifizierung:
   Der Client verbindet sich mit dem Access Point, der eine Authentifizierung anfordert. Dabei wird ein EAP-basiertes Verfahren verwendet. Die Authentifizierungsdaten werden an den RADIUS-Server der Heimatorganisation weitergeleitet. Nach erfolgreicher Authentifizierung wird ein Master Session Key (MSK) erzeugt.
2. Pairwise Master Key Generierung:
   Aus dem MSK wird ein Pairwise Master Key (PMK) abgeleitet (256 Bit). Dieser Schlüssel dient als gemeinsame Basis zwischen Client und Access Point für die weitere Absicherung der Kommunikation.
3. 4-Way Handshake:
   Das 4-Way Handshake findet zwischen dem Client und dem Access Point statt – ohne den RADIUS-Server. Das Ziel dieses Prozesses ist daraus den Pairwise Transient Key (PTK) für die eigentliche Datenverschlüsselung zu erzeugen.

Die Skalierbarkeit ist bei diesem Protokoll sehr hoch und auch für die hohe Anzahl an Nutzern auch absolut notwendig. Eduroam erreichte 2024 8,4 Milliarden Authentifizierungs-Anfragen [50]. Das ist auch einer der Hauptgründe warum eine Integration mit QKD eigentlich nicht möglich ist. Eine Umsetzung mit PSK bzw. mit QKD-Keys, die über dedizierte Hardware in der Regel zwischen P2P Verbindungen erzeugt werden, ist eben durch die Anzahl an Teilnehmern unwartbar und so eine Integration von QKD nicht sinnvoll.

Schicht 6 & 7

Die Darstellungs- und Anwendungsschicht zählen zu den anwendungsorientierten Schichten. Für diese ist es schwierig explizit, wie es im vorherigen beschrieben ist, für jeden dieser Layer gesondert eine Integration von QKD in diesen zu beschreiben. Deshalb werden im Folgenden einzelne Anwendungsprotokolle mit einer Integration von QKD-Keys veranschaulicht: Es wurden einige gängige Anwendungsprotokolle herausgegriffen, wie für Mails (SMTP), Zeitübertragung (PTP), Management (SNMP), Dateiaustausch (SSH) und Medien (ZRTP).

Simple Mail Transfer Protocol (SMTP)

Das Standard-Kommunikationsprotokoll, das für den Versand von E-Mails über das Internet verwendet wird, ist SMTP. Es gehört zur Anwendungsschicht des TCP/IP-Protokolls, wird für gewöhnlich mit TLS verschlüsselt und vorrangig von E-Mail-Clients verwendet. SMTP erlaubt es einem E-Mail-Client oder -Server, sich mit einem SMTP-Server zu verbinden und Nachrichten mittels einer Reihe von textbasierten Anweisungen zu senden. Der SMTP-Server transportiert die E-Mail durch das Netzwerk, bis sie den Mailserver des Empfängers erreicht und dort zugestellt wird.

[53] ist der derzeitige Standard von 2008 zum Versenden von Mails, primär zur Einreichung und Weiterleitung von Mails an die SMTP-Server. Dabei müssen mehrere TCP-Verbindungen, die standardmäßig auf Port 25 eingehen, unterstützt werden. SMTP basiert auf HELO (Hello), MAIL FROM, RCPT (Empfänger), DATA, R(E)SET, V(E)R(I)FY, EXP(A)N(D); HELP, NOOP und QUIT - Befehlen beim Client, die der Server mit kurzen Statuscodes quittiert. Selbst im Standard wird darauf hingewiesen, dass auf der Transportebene keine Integrität oder Authentizität erreicht wird. Eine dringend benötigte Sicherheit muss also durch zusätzliche Ende-zu-Ende-Verschlüsselungs-/ Authentifizierungsverfahren gewährleistet werden. Meist wird SMTP daher mit TLS erweitert: implizites TLS (SMPTS) oder früher explizites TLS (STARTTLS [54]) ohne Verschlüsselung des Vebindungsaufbaus. Seit 2018 wird von der IETF von STARTTLS abgeraten, da ein Downgrade-Zwang vom Server nicht verhindert werden kann, wenn beim Handshake die Unterstützung von STARTTLS nicht angegeben wird. RequireTLS [55] erzwingt eine TLS-Verschlüsselung, allerdings ebenso ein Zertifikat, sonst folgt ein Sendungsabbruch. Bezüglich der Authentifizierung ist eine PKI wegen dem Kettenprinzip der Vertraulichkeit stets schwächer als eine direkte gegenseitige Authentifizierung z.B. via Digitaler Signaturen oder PSKs der Gesprächspartner. Mittlerweile ist es üblich den Extended SMTP (ESMTP) Standard authentifiziert über Port 587 zu verwenden. Dieser ermöglicht SMTP-Auth als Erweiterung und führt auch die STARTTLS-Verschlüsselung ein. Hier wird der frühere HELO durch den EHLO-Befehl ersetzt neben weiteren Parametern zu Befehlen.

Verschlüsselungsprotokolle bei SMTP

Verschiedene eingesetzte explizite Verschlüsselungsprotokolle im Zusammenhang mit SMTP sind OpenPGP, S/MIME und (START)TLS. Diese basieren auf asymmetrischen Verschlüsselungsmethoden, welche erstmal ungeeignet für eine direkte Integration mit QKD sind. Dennoch lassen sich manche Verschlüsselungsprotokolle für SMTP zur Nutzung von QKD-PSKs erweitern. Vorstellbar ist eine Integrationvon QKD bei TLS und OpenPGP, wie im folgenden Abschnitt näher erläutert wird. S/MIME wird hier nicht weiter beschrieben wegen dessen indirekten Schlüsselaustauschverfahren via PKI, wie auch die Verwendung vom BSI abgeraten wird [56]. Auch von STARTTLS wird seit 2018 von der IETF abgeraten und wird daher im Folgenden ebenfalls nicht näher erläutert.

TLS mit SMTP

Wie oben beschrieben wird nur noch implizites TLS mit unmittelbarem Beginn des TLS-Handshakes auf einem expliziten Port empfohlen [57]. Dies wird vom SMTPS unterstützt, welches SMTP mit SSL auf Port 465 kombiniert. In E-Mail-Headern wird eine verschlüsselte SMTPS-Übertragung oft mit dem Übertragungstyp ESMTPS oder ESMTPSA (wenn zusätzlich authentifiziert) gekennzeichnet, wie in [58] beschrieben. Wie im Kapitel zu TLS 1.3 schon angesprochen, ist eine Integration von QKD in TLS aufgrund der Unterstützung von symmetrischen PSKs (RFC 4279 [59]) grundsätzlich möglich.

Und es werden immer mehr zukunftssichere Erweiterungen von TLS eingeführt: Am 4. März 2026 wurde von der IETF ein weiterer Standard für TLS veröffentlicht, der Encrypted Client Hello (ECH) RFC 9849 [60] einführt, bei dem der Handshake verschlüsselt ist und so sensible Session Parameter nicht mehr öffentlich sichtbar sind. Für solche neu eingeführten Erweiterungen ist es wichtig, dass ein Server unbekannte Werte ignoriert, was mit Generate Random Extensions And Sustain Extensibility (GREASE) schon vorher simuliert wird, sodass nun auch Server, die keine ECH-Nachrichten bedienen können, trotzdem nicht durch den ECH-Wert zum Verbindungsabbruch gebracht werden.

OpenPGP

Open Pretty Good Privacy (PGP) [61], aktuell [62] mit neuen kryptographischen Algorithmen, basiert auf einer asymmetrischen Schlüsselarchitektur zum Schlüsselaustausch, die anschließende Verschlüsselung hingegen basiert auf symmetrischer Verschlüsselung. PGP generiert einen zufälligen Session Key mit dem die Nachricht verschlüsselt wird. Nur der Session Key wird mit dem öffentlichen Schlüssel verschlüsselt. Die öffentlichen Schlüssel werden an alle Kommunikationspartner, die verschlüsselte Emails senden sollen, verteilt. Eine gute Praxis ist es, diese an die Signatur von Emails anzuhängen, da man so sicherer sein kann, dass der Schlüssel auch wirklich zu dieser Person gehört [63]. PGP gilt durch den direkten gegenseitigen Schlüsselaustausch als sicherer als S/MIME [64], welche von einer PKI abhängig ist und damit davon, dass stets alle Glieder der Kette auch vertrauenswürdig sind. Bei GNU Privacy Guard (GPG) handelt es sich um eine Open Source Implementierung von OpenPGP.

Zudem bietet PGP/GPG durch seine Unabhängigkeit von einer PKI einen Vorteil bei der Integration in QKD: Der zufällig generierte symmetrische Session Key oder alternativ auch das asymmetrische Schlüsselpaar zur Verschlüsselung des zufälligen Session Keys lässt sich durch ein symmetrisches QKD-Paar ersetzen. Es kann auch komplett auf die asymmetrische Verschlüsselung des Session Keys verzichtet werden: Das Symmetric-Key Encrypted Session Key Packet erlaubt es, dass die Nachricht entweder mit einer Passphrase (hier QKD-Key) oder einem öffentlichen Schlüsselpaar entschlüsselt werden kann. So können Behörden, die bereits QKD-Geräte besitzen, das sichere Schlüsselaustauschnetzwerk auch für sichere symmetrische E-Mail-Verschlüsselung nutzen. GPG könnte so abgeändert werden, dass sich geheime Mails nur mit dem zukunftssicheren symmetrischen QKD-Schlüssel entschlüsseln lassen. Ein mobiles Abrufen der E-Mails vom Server auf dem Smartphone wäre nicht nötig, da die App-Implementierung ohnehin als unsicher gilt, wäre aber im Notfall parallel möglich, wenn der QKD-Key mit PQC-verschlüsselt sicher an weitere Endgeräte übertragen werden würde, solange er nicht im KMS bereits gelöscht wurde. QKD würde zudem den Vorteil mitbringen, dass eine regelmäßige Schlüsselrotation des TLS-Kanals oder bei jeder neuen Nachricht (im Gegensatz zum jetzigen undefinierten Austauschweg der Schlüssel) sehr einfach durch das vorhandene QKD-Netzwerk ermöglicht werden würde.

Folgende Forschungsarbeit [65] zeigt, wie QKD-Keys exemplarisch in ein OpenPGP-basiertes Chat-Tool integriert wurden:

Für die Integration von QKD in SMTP wurde zwischen der Verschlüsselung des Kommunikationskanals oder der eigentlichen Nachricht unterschieden. Für die Verschlüsselung der Nachricht an sich gilt

(Layer 6), dass jede E-Mail die versendet wird, einen QKD-Schlüssel vom jeweiligen KMS erfordert (vgl. Abbildung 13). Dieser wird zur symmetrischen Verschlüsselung des Textinhalts der Nachricht verwendet. Die eindeutige Schlüssel-ID wird gespeichert und im Header übertragen, damit der empfangende SMTP-Server den entsprechenden QKD-Schlüssel aus seinem KMS abrufen kann. Die Nachrichten können hierbei zum Beispiel über die Python Bibliothek Fernet verschlüsselt werden [66], die lediglich den geheimen Schlüssel, die Nachricht und einen Zeitstempel als Eingaben benötigt, dabei wird der Eingabeschlüssel (256 Bit) automatisch innerhalb dieser Bibliothek aufgespaltet:

- 128 Bit werden für die reine Verschlüsselung mit AES verwendet
- 128 Bit werden für die Authentifizierung mit einer HMAC-SHA-256-Signatur verwendet

Es wird eine KDF aus einem klassischen Schlüssel oder PQC-Schlüssel mit einem QKD-Schlüssel erstellt. In Fernet gibt es dazu die verschiedenen KDFs: PBKDF2HMAC, Argon2id oder Scrypt. Hier können mit derive(key_material) PSKs als Schlüsselmaterial berücksichtigt werden. Dieser verschmolzene Schlüssel wird dann zur symmetrischen Verschlüsselung der E-Mails nur zwischen den Servern eingesetzt und am Empfangsserver sofort QKD-dekodiert, sodass die Mail im Postfach nur noch klassisch oder PQC-verschlüsselt verbleibt.

Die Entschlüsselung der Mails erfolgt also sofort durch den empfangenden Mailserver. Daher ist eine Synchronisierung der QKD-Schlüssel zwischen mehreren Geräten nicht notwendig, auch wenn die Benutzer die Messaging-Anwendung auf mehreren Geräten verwenden. Es muss jedoch beachtet werden, dass die Schlüsselverwaltung nicht trivial ist, wenn E-Mails auf mehreren Geräten gelesen werden müssen, kann der Schlüssel im KMS nicht mehr verfügbar sein, weil er von einem anderen Gerät des Benutzers abgerufen und danach gelöscht wurde oder wenn der Empfänger nicht online ist, da er die verschlüsselte Nachricht zu einem späteren Zeitpunkt öffnet und so der Schlüssel im vorhandenen KMS nicht mehr verfügbar sein kann. Daher ist ein durchaus komplexes KMS für dieses Vorgehen nötig, das vor allem eine dynamische Schlüssel-Lebensdauer verwaltet, automatische Schlüsselrotation ermöglicht und eine sichere Verteilung der Schlüssel an die beteiligten Endpunkte sicherstellt.

Die durchschnittliche Schlüsselraten von QKD-Geräten haben eine Geschwindigkeit von circa 1-2 kbit/s. Aus diesem Grund ist genau zu überlegen, ob der SMTP-Server jede Nachricht mit einem eigenen QKD-Schlüssel verschlüsseln oder einige Nachrichten in größeren Mengen übermitteln soll, um die Anzahl der benötigten Schlüssel zu verringern [65].

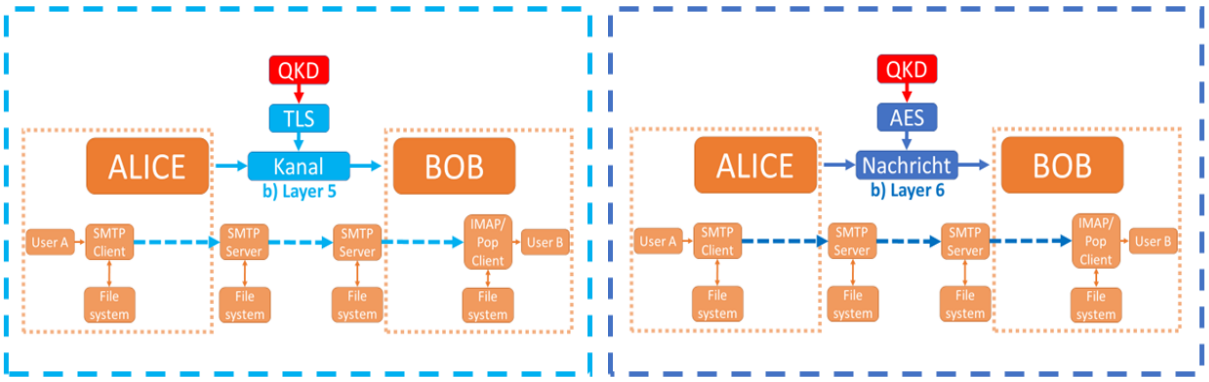


Abbildung 13: Mögliche Integrationen von QKD in SMTP [65] : a) Es gibt einmal die Möglichkeit zur Verschlüsselung und Authentifizierung des Kanals mit TLS vgl. Kapitel TLS 1.3 (Layer 5). b) Desweiteren gibt es die Möglichkeit zur Verschlüsselung der eigentlichen Nachricht mit AES, in dem QKD-Keys als Preshared Keys (PSKs) bereitgestellt werden (Layer 6). Dabei können diese beiden Verschlüsselungsverfahren parallel oder jeweils einzeln benutzt werden.

Authentifizierung bei SMTP

Zunächst muss zwischen der Authentifizierung des Zugriffs auf den SMTP-Server und der Authentifizierung des Absenders unterschieden werden. Zur Authentifizierung des Absenders können Digitale Signaturen (Signierungsschlüssel sind stets asymmetrisch, da sonst die Signatur vom Empfänger verändert werden könnte) oder auch PGP oder S/MIME eingesetzt werden. Davon unterscheiden sich die Server-Authentifizierungsverfahren, wie SMTP-Auth oder Authentifizierung über mTLS/Client-Zertifikate. Diese stellen sicher, dass nur autorisierte Clients Mails an den Server versenden dürfen. Die klassische passwortbasierte Authentifizierung (Basic Auth) wird von großen Providern zunehmend abgeschaltet. PSKs gelten in der klassischen Kryptographie als weniger sicher als Zertifikate, da der geheime Schlüssel auf beiden Seiten im Klartext (oder reversibel verschlüsselt) vorliegen muss. Dennoch stellt das für ein externes QKD-Netz mit sicheren Schlüsselpuffern keine Herausforderung dar.

SMTP-Auth [67] ist eine Erweiterung von ESMTP. SMTP-Auth nutzt Simple Authentication and Security Layer (SASL), um verschiedene Authentifizierungsmechanismen zu unterstützen. Nach dem EHLO-Befehl wird AUTH eingeleitet. Zu den unterstützten Authentifizierungsmethoden gehören u.a.:

- PLAIN / LOGIN: Überträgt Benutzername und Passwort Base64-kodiert. Diese Encodierung ist nur ein anderes Darstellungsformat und sollte daher nur innerhalb einer verschlüsselten TLS-Verbindung erfolgen.
- CRAM-MD5 / DIGEST-MD5: Ein Challenge-Response-Verfahren, bei dem der Server eine Aufgabe sendet, die der Client mit seinem Passwort löst. Somit wird das Passwort überprüft, ohne es selbst zu übertragen.
- OAuth2: Eine neuere Methode, bei der statt eines Passworts ein zeitlich begrenztes Token verwendet wird. Dies wird heutzutage bei Google und Microsoft standardmäßig eingesetzt.

Je nach Ausgang der Verifizierung, wird der Mailversand gestartet oder abgebrochen.

Der Standard zur E-Mail-Einreichung [68] von 2011 bringt sogar eine Authentifizierungspflicht mit sich. Unter bestimmten Fällen kann der Server auch auf eine Authentifizierung verzichten, wie nach einer kürzlich vorangegangenen Anmeldung via POP zum Abrufen der Mails mit derselben IP-Adresse.

QKD-Schlüssel zur PSK-Authentifizierung könnten sowohl bei PLAIN als Passwort eingesetzt werden, als auch als Passwort für den Challenge-Response Mechanismus. Beim sichereren OAUth 2.0 können als Secrets ebenfalls QKD-Keys eingesetzt werden, um temporäre Tokens zu generieren. Wenn man von den gängigen Architekturen abweicht, kann man auch ausschließlich direkt QKD-PSKs einsetzen, um sich beim Server, der an das QKD-Netzwerk angeschlossen ist, zu authentifizieren.

Precision Time Protocol (PTP) und White Rabbit (WR)

Das Precision Time Protocol (PTP) nach IEEE 1588 ist ein Netzwerkprotokoll, das zur hochpräzisen Synchronisation von Uhren im Mikrosekunden bzw. Nanosekundenbereich in verteilten Systemen dient. Damit ist PTP deutlich präziser als herkömmliche Protokolle wie das Network Time Protocol (NTP), dass nur eine Genauigkeit im Millisekundenbereich erreicht. PTP arbeitet nach einem Master-Slave-Prinzip. Dabei übernimmt eine sogenannte Master Clock die Rolle der Zeitreferenz und gibt ihre Zeit an mehrere Slave Clocks weiter. Die Synchronisation erfolgt über den Austausch spezieller Nachrichten, wie in Abbildung 14 gezeigt und beschrieben [69].

Für einige Anwendungen reicht jedoch diese Zeitsynchronisation im Mikrosekunden bzw. Nanosekundenbereich nicht aus und es wurde eine Erweiterung des PTP Protokoll entwickelt, dass

deutlich höhere Zeitgenauigkeiten im Sub-Nanosekundenbereich gewährleistet: Das White Rabbit (WR) PTP-Protokoll von CERN. Dabei wird die Ungenauigkeit, die durch die Asymmetrie des physikalischen Mediums (z.B. unterschiedliche Ausbreitungsgeschwindigkeiten) oder allgemein die Hardware-Schnittstellen (z.B. unterschiedliche Verbindungslängen) verursacht wird, durch eine geeignete Vorabmessung ermittelt und in der Synchronisation mit berücksichtigt. Ein großer Nachteil hierbei ist, dass man geeignete Hardware und so auch einen deutlich höheren Implementierungsaufwand durch Vorabmessungen hat. Während PTP - IEEE 1588 universell einsetzbar und weit verbreitet ist, bleibt White Rabbit eine spezialisierte Lösung für extreme Präzisionsanforderungen [70].

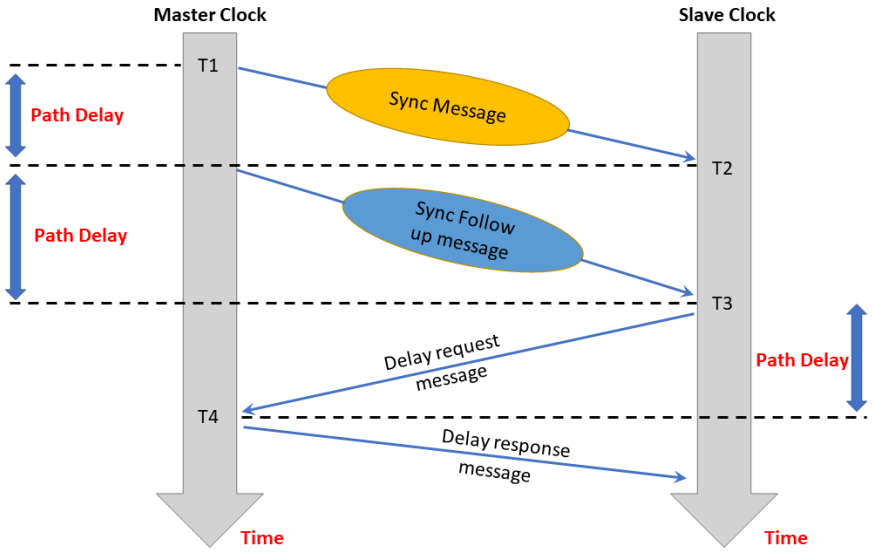


Abbildung 14: Die Protokollnachrichten umfassen die Hauptnachricht für die Zeitsynchronisierung (Sync-message), die Sync Follow up message (Zweischrittmodus), die sekundäre Verzögerungsanforderungsnachricht (Delay request message) sowie die Antwort der Master Clock auf die Verzögerung. Der Master überträgt seine Zeitansicht mittels sync-messages an die Slaves; jedes Netzelement entlang des Übertragungswegs empfängt die Sync-Nachricht und fügt ihr eine Korrektur hinzu. Die Korrektur umfasst die gemessene Übertragungsverzögerung des Eingangsports, an dem die Sync-Nachricht empfangen wurde. Der Master und der Slave senden IEEE 1588-Nachrichten mit den Bezeichnungen DELAY REQUEST und DELAY RESPONSE zwischen einander, um die Verzögerung zu messen [69].

Zeitinformationen stellen einen essenziellen Bestandteil moderner informationsverarbeitender Systeme dar. Sie werden unter anderem zur Ereignisprotokollierung, zur Synchronisation verteilter Systeme sowie zur Gewährleistung von Integrität und Nachvollziehbarkeit eingesetzt. Gleichzeitig können Zeitdaten als sensitive Metainformationen fungieren, da sie Rückschlüsse auf Systemzustände, Kommunikationsmuster oder Nutzerverhalten ermöglichen. Vor diesem Hintergrund ist der Schutz von Zeitinformationen von besonderer sicherheitstechnischer Relevanz.

Mit QKD ist es möglich, solche Zeitinformationen bei einer WR-PTP Übertragung (nach Standard IEEE 1588 von 2019 [71]) zu verschlüsseln. Die Struktur dieser Übertragung ist in Abbildung 15 gezeigt. Bei dieser Übertragung teilen sich Seite A und B sowohl die Zeit- als auch Frequenzreferenz, die durch das WR-PTP-Protokoll kodiert werden. Das WR-PTP-Gerät A, das an eine externe Frequenzreferenz angeschlossen ist, ändert Phasenschwankungen der Frequenz in zeitliche Verzögerungen der WR-PTP-Nachrichten, die zwischen den Knoten A und B ausgetauscht werden. Ein großer Vorteil ist, dass es hierbei eigentlich unmöglich ist die vorhandene Zeitinformation als Unbefugter abzuhören. Wird das Frequenzsignal um eine zufällige Phase verschoben, erfährt das Zeitinformationssignal in A und B eine

zufällige zeitliche Verzögerung, wodurch B keine Möglichkeit hat, eine Schätzung der Zeitinformation in A vorzunehmen. Um eine geheime zufällige Phase mit QKD zu erzeugen, müssen sich A und B die Schlüssel in einem lokalen KMS, das sich jeweils auf den entsprechenden Seiten befindet, teilen. In A wählt der User einen Schlüssel aus, erzeugt mit diesem die Phasenverschiebung nach Gleichung (1), bringt sie auf das Zeit-Signal an und überträgt das codierte Zeitsignal an B. B kann die Phasenverschiebung lokal ausgleichen, indem B einen entgegengesetzten Phasenwert mit dem gleichen Schlüssel hinzufügt und dadurch ein Zeitsignal erzeugt, das mit A synchronisiert ist.

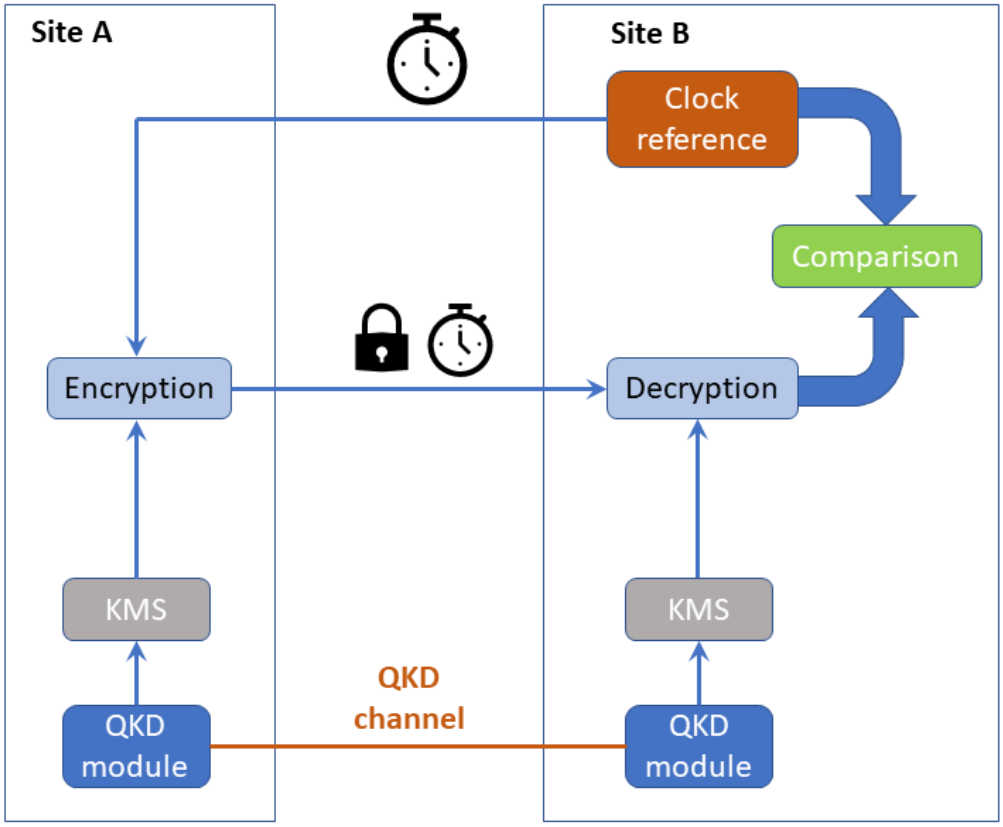


Abbildung 15: Schematische Darstellung von WR-PTP verschlüsselt mit QKD: Eine Zeitreferenz von Standort B wird über WR-PTP nach Standort A übertragen, dort wird eine Phasenverschiebung auf dem Zeitsignal basierend auf dem QKD-generierten Schlüssel nach Gleichung (1) *erzeugt und dieses Signal zurück nach B gesendet. Am Standort B kann diese Phasenverschiebung mit dem zugehörigen QKD-Key lokal ausgeglichen werden, sodass A mit B synchronisiert ist [72].*

Die QKD-Schlüssel K werden als zufällige hexadezimal Zahlen, die in einem Array zusammengefasst sind, ausgegeben. Danach werden diese zufälligen hexadezimal Zahlen als *N* Paare angeordnet und in das Dezimal System überführt ($K = \{k_1, \ldots, k_{2N}\}$). Es ergibt sich folgende Gleichung (1) für Berechnung der zufälligen Phase Φ, die für die Zeitsynchronisation notwendig ist:

$$\Phi = \frac{k_{2i}k_{2i+1}}{C}, \qquad i \in [0, N-1] \tag{1}$$

C ist eine konstante ganze Zahl, die verwendet wird, um das hinzugefügte weiße Phasenrauschen auf einen Maximalwert unterhalb von 2π zu beschränken [72].

Simple Network Management Protocol (SNMP)

Das Simple Network Management Protocol (SNMP) ist ein standardisiertes Protokoll zur Überwachung und Verwaltung von Netzwerkkomponenten wie Routern, Switches, Servern oder Druckern. Es ermöglicht die zentrale Erfassung von Statusinformationen sowie das Senden von Konfigurationsbefehlen an Netzwerkgeräte. SNMP ist ein elementarer Bestandteil moderner Netzwerkinfrastrukturen und wird in nahezu allen größeren Netzwerken eingesetzt. Dieses Protokoll dient dem Austausch von Verwaltungsinformationen zwischen Manager und Agent. Ein SNMP-Manager kann einen SNMP-Agenten steuern, indem er SNMP-Nachrichten austauscht. Jeder SNMP-Agent hat eine Datenbank mit Variablen, die als MIB (Management Information Base) bezeichnet wird. Mit dieser Datenbank fragt der Manager den SNMP-Agenten an und deutet die empfangenen Signale. Agenten sind Softwaremodule, die auf den verwalteten Geräten installiert sind. Sie reagieren auf Anfragen des Managers, indem sie Informationen aus ihrer MIB nutzen und über das SNMP-Protokoll diese an den Manager senden [73].

SNMP wurde in mehreren Versionen weiterentwickelt, wobei sich insbesondere in Bezug auf Sicherheit und Authentifizierung deutliche Unterschiede ergeben:

- **SNMPv1** (1988) war die erste Version und setzte auf ein einfaches Community-String-Modell zur Authentifizierung, das keine Verschlüsselung bot. Es war funktional, aber aus heutiger Sicht äußerst unsicher.
- **SNMPv2** brachte einige Erweiterungen in der Funktionalität und Performance (z. B. Bulk Transfers), übernahm jedoch das gleiche unsichere Authentifizierungsmodell wie SNMPv1 – ebenfalls nur mit Community-Strings und ohne echte Zugriffskontrolle oder Verschlüsselung.
- **SNMPv3** stellt einen Meilenstein dar, da hier erstmals umfassende **Sicherheitsmechanismen** eingeführt wurden. Im Gegensatz zu den Vorgängerversionen bietet SNMPv3:
    - **Authentifizierung** (z. B. mittels HMAC mit SHA oder MD5), um die Identität der Kommunikationspartner zu verifizieren,
    - **Verschlüsselung** (z. B. mit DES oder AES), um die Vertraulichkeit der übertragenen Daten zu gewährleisten,
    - sowie eine **detaillierte Zugriffskontrolle**, mit der sich genau definieren lässt, welche Benutzer auf welche Daten zugreifen dürfen.

Diese Verbesserungen machen SNMPv3 zur bevorzugten Wahl in sicherheitskritischen Umgebungen. Es ist die einzige Version, die heutigen Anforderungen an Datenschutz und Integrität gerecht wird und womit eine Verschlüsselung mit QKD überhaupt Sinn macht.

SNMPv3 kennt drei Sicherheitsstufen, die in folgender Tabelle zusammengefasst werden:

Sicherheitsstufe	**Bezeichnung**	**Schutz**	**Benötigte Paramter**
noAuthNoPriv	Kein Schutz	Keine Auth., keine Verschlüsselung	Nur Benutzername
authNoPriv	Authentifizierung	Schutz vor Manipulation	Benutzername + authKey
authPriv	Auth + Verschl.	Schutz vor Manipulation und Abhören	Benutzername + auth- **und** priv-Key

Die Integration von QKD in SNMPv3 ist grundsätzlich denkbar, allerdings nicht ohne weiteres im Protokoll vorgesehen. SNMPv3 arbeitet mit Passphrasen (PSK), aus denen intern kryptografische Schlüssel abgeleitet werden. QKD hingegen erzeugt fertige symmetrische Schlüssel, die zwischen zwei Endpunkten abhörsicher übertragen werden. Um diese unterschiedlichen Konzepte miteinander zu verbinden, lassen sich zwei Integrationsansätze unterscheiden:

In dieser Variante wird ein über QKD erzeugter Schlüssel als neues Authentifizierungs- oder Verschlüsselungspasswort in SNMPv3 verwendet, vergleichbar wie beim Erstellen eines neuen W-LAN Passworts. Da SNMPv3 aus diesen Passphrasen intern die tatsächlichen Schlüssel (authKey und privKey) ableitet, können QKD-Schlüssel sozusagen als „starkes Preshared Secret“ genutzt werden. Der QKD-Schlüssel müsste dabei sowohl auf dem SNMP-Manager als auch auf dem Agenten synchron eingespielt werden, entweder manuell oder automatisiert. Dies ist technisch relativ einfach, allerdings fehlt dabei die Möglichkeit, Schlüssel zur Laufzeit dynamisch zu wechseln. Die Integration bleibt statisch und erfordert eine koordinierte Umstellung auf beiden Seiten, was im täglichen Betrieb sehr aufwendig sein kann.

Ein fortgeschrittenerer Ansatz wäre, die durch QKD erzeugten Schlüssel direkt als authKey und privKey in den SNMPv3-Stack einzuspeisen – ohne sie zuvor aus einer Passphrase abzuleiten. Technisch würde das bedeuten, dass das SNMP-System die kryptografisch sicheren Schlüssel direkt aus dem QKD-Key-Management-System übernimmt und sie für Authentifizierung und Verschlüsselung verwendet. Diese Methode wäre deutlich dynamischer und würde dem Prinzip der QKD-Schlüsselrotation besser gerecht. In der Praxis ist sie allerdings sehr komplex, da SNMPv3 typischerweise keine externe Schlüsselzufuhr vorsieht. Eine direkte Key Einspeisung setzt daher voraus, dass der SNMP-Stack erweitert oder angepasst wird, z. B. durch Zugriff auf interne APIs oder die Verwendung speziell angepasster SNMP-Agenten und -Manager. Auch die Schlüsselverteilung und Synchronisation muss sicher und fehlerfrei erfolgen, was ein durchdachtes Systemdesign erfordert.

Secure Shell (SSH)

SSH-1 mit Secure Copy (SCP) zur Dateiübertragung bzw. sein Nachfolger SSH-2 mit SSH File Transfer Protocol (SFTP) ermöglicht eine bisher integre, authentifizierte und verschlüsselte Remote-Datenübertragung zwischen Computern über die lokale Konsole. SSH besteht aus dem Transport Layer Protocol, in dem eine asymmetrische Verschlüsselung zur initialen Verschlüsselung der Sitzung für die Verhandlung über die anschließende symmetrische Verschlüsselung erfolgt. Auch für die anschließende gegenseitige Authentifizierung wird meist asymmetrische Verschlüsselung genutzt. Darauf aufbauend regelt das Connection Protocol die einzelnen Verbindungen über den verschlüsselten Gesamtkanal. Die eigentliche Verschlüsselung erfolgt symmetrisch. Vor der Authentifizierung erfolgt die Aushandlung des verwendeten Verschlüsselungsverfahrens. AES128/265 (GCM oder zusätzlich HMAC zur Authentifizierung) als Verschlüsselungsverfahren und ECDH & DH als Methode zum Schlüsselaustausch werden vom BSI nur noch bis 2031 empfohlen [74]. Für die zukünftig sichere Übertragung in der Shell wird maßgeblich auf PQC gesetzt, wie im IETF-Draft [75]. Danach implementiert OpenSSH 10.0 schon jetzt den 2024 NIST-standardisierten Module-Lattice-Based Key-Encapsulation Mechanism hybrid mit ECDH: mlkem768x25519-sha256. Dennoch ist eine SSH-Verbindung und damit der kontrollierende Fernzugriff auf ein System sehr sicherheitskritisch und es bietet sich zwischen Rechenzentren an, diesen Zugriff mit QKD-Keys abzusichern. Während eine symmetrische Verschlüsselung einen zeitlichen Vorteil mitbringt, da der Rechenaufwand für asymmetrische Verschlüsselung größer ist, bietet QKD als eigenes *Out-of-band* Schlüsselnetzwerk mit Puffern eine schnelle Schlüsselbereitstellung unter Voraussetzung von schnellen Schnittstellen und Routing.

[76] verwendet daher das bekannte QKD-Protokoll BB84 zusammen mit SSH. Nach einer erfolgreichen klassischen Authentifizierung, wird im Handshake der klassische Kanal nur noch zur Parameteraushandlung für BB84 genutzt. In der anschließenden SSH-Verschlüsselung mit symmetrischen Schlüsseln können somit QKD-Schlüssel eingesetzt werden.

Anzudenken wäre ebenfalls, ob die initiale asymmetrische DH-Schlüsselaushandlung durch die sofortige Nutzung von QKD-Keys als PSKs ersetzt werden könnte. Hier könnten die asymmetrischen Schlüssel zur Verschlüsselung der geheimen Schlüssel durch symmetrische QKD-Keys ersetzt werden. Dazu müssten, wie Abbildung 16 zeigt, die Algorithm Negotiations angepasst werden, sodass sich auf QKD-PSKs geeinigt werden kann, sowie die Key Generation und sich daraus ergebende Key Derivations. Da als Authentifzierungsmethoden nur asymmetrische Schlüssel, (Host-basierte) oder Passwort-basierte (zu langsames Rekeying) Methoden vorgeschlagen werden, eignet sich der Einsatz von QKD-Keys ohne tiefgreifende Protokolländerungen /-ersetzungen nicht für den Schlüssel des User Authentication Protocols.

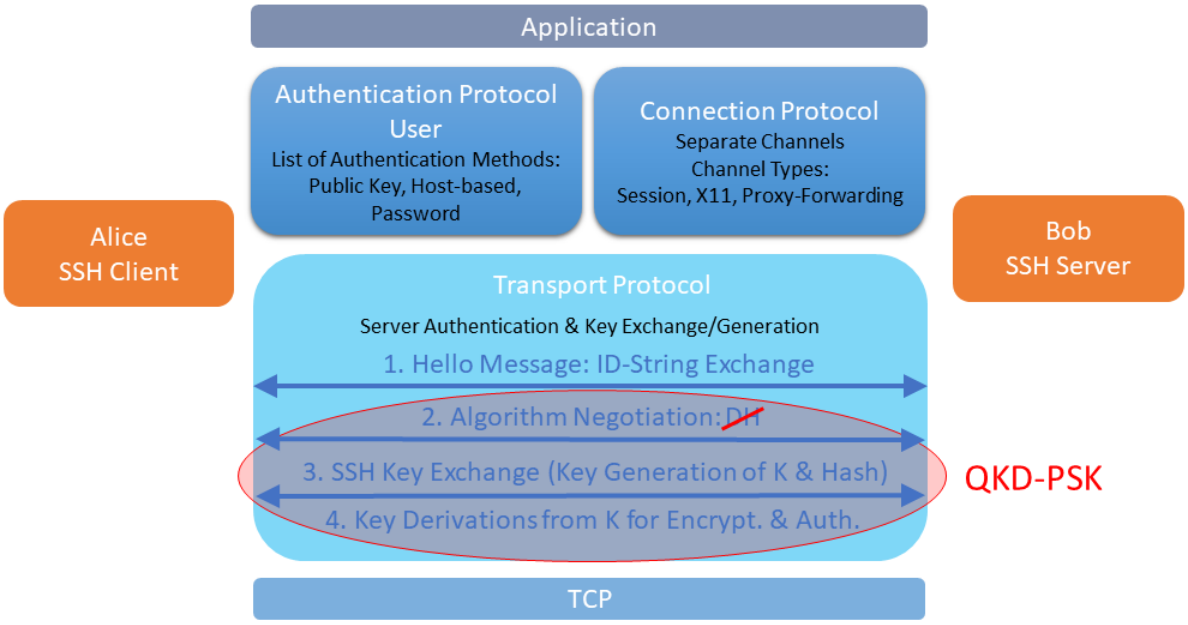


Abbildung 16: SSH-Protokoll mit QKD: Der rote Kreis markiert Einsatzmöglichkeiten von QKD: zwischen der TCP-Schicht und der Applikationsschicht befinden sich drei SSH-bezogene Protokolle: für die Integration von QKD zu Verschlüsselung ist das Transport-Protokoll mit seinem Handshake-Protokoll von Bedeutung. In dieser Verhandlung soll anstatt DH ein quantensicherer PSK aus QKD genommen werden. Somit werden weitere DH-basierte Schlüsselberechnungen redundant.

Laut ETSI [77] hängt aber die Integration von QKD in die Verschlüsselung von SSH maßgeblich von der benötigten sicheren Schlüsselrate im Vergleich zur verfügbaren Rate an QKD-Schlüsseln ab. Im RFC 4253 [78] wird empfohlen, nach jedem GB Daten bzw. nach einer Stunde ein Rekeying durchzuführen. Wichtig ist, dass alle Rechner auch quanten-sichere Protokolle unterstützen, damit es nicht durch Proxy-Forwarding zur Sicherheitsbeeinträchtigung des gesamten Netzwerkes kommt.

Mediendateien mit echtzeitkritischen Protokollen

Für den Empfang von Video- und Audiodateien in Echtzeit, bedarf es spezieller Protokolle mit niedriger Latenz, wie in Schicht 4 & 5 schon deutlich wurde. Das proprietäre Real Time Messaging Protocol (RTMP) für Flash-Player agiert direkt auf Basis vom TCP/IP-Port 1935, mit *RTMPT* auf Basis von HTTP und *RTMPS* auf Basis von HTTPS für eine verschlüsselte Verbindungen in SSL/TLS. Video-Streaming-Anwendungen benötigen jeweils ihren eigenen Server für das Senden und einen für den Empfang der sensiblen Daten in Echtzeit. [79] zeigt, wie das RTMP-Protokoll für Video-Streaming mit QKD-Keys

eingesetzt wird. Dazu muss die Payload in verschiedene Fragmente zerteilt werden, da ein Fragment nur 128 Bytes (Video) bzw. 64 Bytes (Audio) fassen kann. Die Übertragung der RTMP-Daten zum Zielserver erfolgt aber über einen HTTP-Tunnel (RTMPT). Echtzeit-Videokonferenzsysteme sind meist IP-basiert und können bei ausreichender SKR nur mit OTP verschlüsselt sein. Für Videoanwendung wird unter anderen auch das Real-Time Streaming Protocol (RTSP) genutzt. Im Gegensatz zu den meisten anderen Videoprotokollen ist dieses nicht direkt mit HTTP kompatibel, sodass es nicht für Webbrowser, sondern für die private Übertragung von Videoströmen z.B. innerhalb von Rechenzentren geeignet ist. Wegen seiner geringen Popularität besteht hier leider noch keine Literatur über eine QKD-Integration damit.
Während RTMP und RTSP an sich erstmal unverschlüsselt sind, bringt das häufig eingesetzte HTTP Live Streaming (HLS)-Protokoll Verschlüsselung auf Kosten von Latenz mit. Dagegen bietet das Secure Reliable Transport (SRT)-Protokoll, welches das UDP einsetzt, Ende-zu-Ende-Verschlüsselung mit AES 128/256 und geringere Latenz als RTMP. Hier könnten einfach zur Verfügung stehende *Out-of-band* QKD-Keys stattdessen eingesetzt werden, um dieses Open-Source Protokoll quantensicher zu machen. Zusätzlich sollte ein Echtzeit-Monitoring vom QKD-Netzwerk, wie in [80], eingesetzt werden, auch um die Schlüsselrate gerade beim hybriden Einsatz von OTP zu überwachen und rechtzeitig auf AES-QKD bzw. auf klassische Schlüssel umzusteigen. Ein weiteres häufig genutztes Protokoll zur Enkryption mit AES bzw. TwoFish 128/192/256 für Medienanwendungen mit Audiostream ist das RTP von Zimmermann (ZRTP) [81], welches für das Secure Real-time Transport Protocol (SRTP) [82] die Schlüsselaushandlung übernimmt. ZRTP wird für Unicast VoIP oder Sprachkanal (auch Video) verwendet und bringt folgende Merkmale mit sich:

- Nur für Punkt-zu-Punkt-Verbindungen
- Forward Secrecy, da DH Keys nach jedem Anruf gelöscht werden
- Mit dem Session Initiation Protocol (SIP) wird ein RTP-Strom aufgebaut: eigene Schlüsselgenerierung mit Salt für SRTP Session Keys & Parameter
- Unabhängige Schlüsselaushandlung von RTP direkt über den Medienpfad
- Keine PKI oder andere dritte Instanz zur Authentifizierung zwingend notwendig
- Als Public Key Algorithmus wird dennoch meist Ephemeral DH verwendet
- Hello-Messages sind von Beginn an mit HMAC gekennzeichnet zur Verhinderung von Replay-Attacken
- Optional: Digitale Signatur zur Signierung der SAS (Übermittlung in den verschlüsselten Confirm-Messages)
- Short Authentication String (SAS) wird angezeigt zur verbalen Überprüfung (extra Feature zur manuellen Authentifizierung)
- Key Continuity: Ein kleiner Anteil Schlüsselmaterial in Form von einem vor der Löschung berechnetem MAC geht in den nächsten Schlüssel ein, sodass eine Autorisierung vom System möglich ist

Bei ZRTP sollte die PKI mit Ephemeral DH durch QKD PSK ersetzt werden. Auch in SIP, welches die Signalverbindung handhabt, kann bei ZRTP für den Schlüsselaustausch quantensicher aufgerüstet werden, indem von vornherein QKD-Keys eingesetzt werden. Zusätzlich lässt sich der SAS durch einen QKD-Key ersetzen, was eine manuelle Überprüfung ersetzen könnte. So lässt sich die anschließende SRTP-Verschlüsselung quantensicher machen.

Während der Hello-Message wird ein 96-Bit-langer zufälliger Identifier (ZID) generiert. Als Entropiequelle soll das Mikrofon dienen. Zusätzlich werden die verfügbaren Ciphersuites in der Hello-Message als Optionen mitgesandt. Einen Ablauf von ZRTP zeigt Abbildung 17. Es werden hier auch

zuerst initiale Handshake-Sitzungsschlüssel und danach erst die Applikations-Sitzungsschlüssel für die weitere Verwendung in SRTP generiert. Es können auch zwei *Out-of-band* PSKs zusätzlich bei ZRTP geteilt werden. Dieses Protokoll wird in der GNU-Implementierung ZRTP4J auch bei der Open-Source Videokonferenzplattform Jitsi in der Praxis angewendet [83]. Ein quantensicheres ZRTP wurde bereits mit PQC-Keys in [84] entwickelt. Es wurde der Schlüsselaustausch durch einen (hybriden) KEM ersetzt, ebenso wie die Sitzungsschlüssel und die Signatur wird entsprechend dem Anfragenden / Antwortenden gelabelt. Hierzu können die verfügbaren Ciphersuites in der Hello-Message angepasst werden. Das Protokoll für den Einsatz von symmetrischen QKD-Keys lässt sich ähnlich abändern. Als sehr nützlich für den Einsatz von QKD-Keys könnten sich die *Out-of-band* PSKs (Shared Secrets s1, s2 (Auxsecret), s3 (PBXsecret)) erweisen. Es könnte sich auch der Pre-shared Mode nutzen lassen, um die DH-Berechnungen (normalerweise nur bei manchen Sessions mit dem vorherigen Schlüssel) ganz zu umgehen, zugunsten eines QKD-PSKs, welcher anstelle des ZRTP MSK in die KDF für den Sitzungsschlüssel einfließen kann. Zu beachten ist, dass dieser Modus nicht von potentiellen Angreifern ausgenutzt wird. Folglich darf für Audio/Video-Streams und für die zusätzliche Sitzungseinwahl nur der Multistream-Modus verwendet werden.

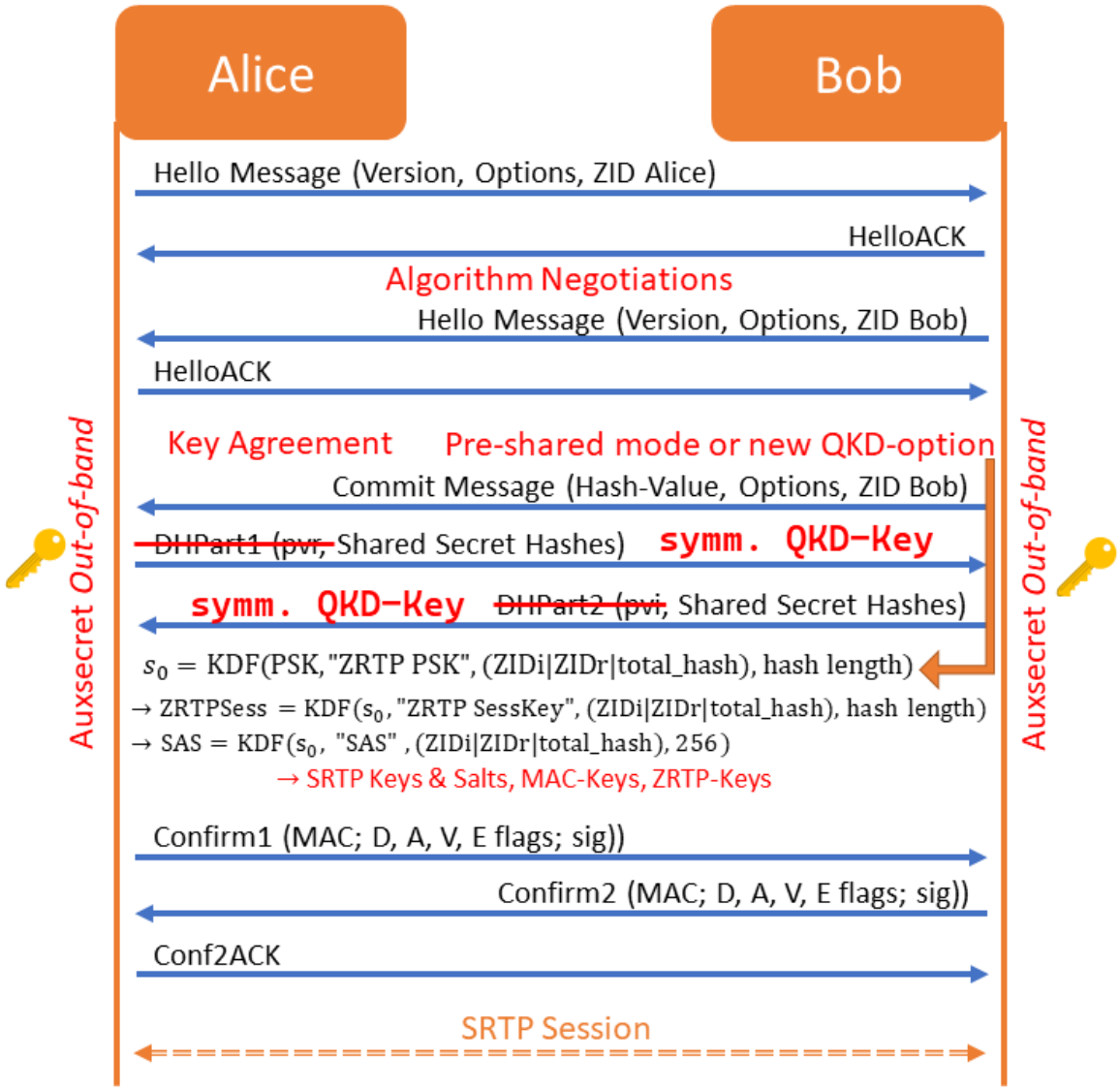


Abbildung 17: Ablauf ZRTP zur Etablierung einer SRTP-Session mit QKD [81]: Hello Message mit Version, Optionen, ZID (ZRTP ID) und Bestätigung (ACK) zwischen den Kommunikationspartnern Alice und Bob; QKD sollte als Algorithmus ausgehandelt werden, sodass beim nachfolgenden Key-Agreement bei DH der symm. QKD-Key entweder als Auxsecret zusätzlich in die Schlüsselableitung zur Absicherung eingehen kann; oder alternativ direkt bei der Key-Agreement Methode den Pre-shared

Mode zu wählen, welcher dann ermöglicht, dass nur der symm. QKD-Key statt DH direkt in die Schlüsselberechnung s0 eingeht (siehe gewinkelter Pfeil); darauf folgt die Generierung der Session Keys und des SAS; verschiedene Flags bedeuten in der Confirm-Message: Flag Disclosure (D), Allow Clear (A), SAS Verified (V) und Private Branch Exchange (PBX)-Enrollment Flag (E); nun beginnt die SRTP-Sitzung mit den ausgehandelten Schlüsseln.

Die bisherigen Algorithmen zur Bildverschlüsselung basieren auf klassischen Algorithmen, die die Bilder/Videoframes mit z.B. sog. *Chaotic Functions* sicher und schnell randomisieren können, bevor sie komprimiert werden (teilweise ohne den Schlüssel zu kennen) [85]. Übliche Verfahren zur Bildverschlüsselung besitzen ihre eigenen zufälligen Verwürfelungen und Transformationen auf Basis von Funktionsparametern und brauchen somit nicht zwingend einen *Out-of-band* Quantenschlüssel für deren Berechnungen. Die *pseudo-random Seeds* könnten aber auch durch QKD-Schlüssel ersetzt werden. QKD wäre aber auch eine Bereicherung für die bisherigen Protokolle, könnten sie gleich auf echte zufällige Bits für das Mapping zurückgreifen. Dies wird in [86] mit der Kombination von BB84 generierten Schlüsseln als Seeds für die mehrdimensionalen hyperchaotischen Funktionen vorgeschlagen. In [87] wird eine hybride Verschlüsselung einer *Block Diagonal Chaotic* (BDC) Matrix (für chaotische Verschlüsselung mit anschließendem Quantum Random Walk) zusammen mit QKD-Keys für die Quantenverschlüsselung der Seeds vorgeschlagen. Manche Verfahren gehen dazu über, *Quantum Image Representation* (QIR) zu nutzen, bei der digitale Bilder mit quantenbasierter Informationsverarbeitung kombiniert werden, wobei diese in Quantenzustände in Quantenschaltkreisen codiert werden. Diese Repräsentation kann auch zur quantenbasierten Verschlüsselung genutzt werden mit anschließender TLS-Übertragung, wie in [88]. Statt dem binären Pseudo-Zufallsschlüssel könnten auch echte zufällige und abhörsichere QKD-generierte Schlüssel eingesetzt werden.

Zeitbezogen kann keine allgemeine Aussage getroffen werden, ob QKD-Schlüsselbereitstellung schneller/langsamer ist, als die rechenintensiven klassischen/PQC-Methoden. Dies hängt immer ab von der konkreten Implementierung des QKD-Netzwerkes in der Praxis: physikalische Distanzen zwischen dem Server und den QKD-Nodes, das Netzwerk und dessen Routing, sowie die spezifische ETSI API Implementierung und vorgefüllte Key-Pools sind alles Einflussfaktoren, die für echtzeitkritische Anwendungen eine Rolle spielen.

Fazit zur Protokollintegration von QKD

Am Häufigsten findet man eine Verschlüsselung auf Schicht 2 und 3 vor. Dabei bietet die MACsec-Verschlüsselung Vorteile in der Latenz, während bedingt durch den größeren Overhead in IPsec die Latenz höher ausfällt. Generell gilt: je weiter unten der Layer, desto weniger Overhead und damit Latenz. Die verschlüsselten Pakete sind jeweils für die höheren Layer transparent. Dafür wird bei IPsec eine einmalige Ende-zu-Ende-Verschlüsselung erzielt, im Gegensatz zu den jeweils einzeln verschlüsselten Punkt-zu-Punkt-Verbindungen von MACsec. Eine Verschlüsselung auf Layer 1 mit OTNsec wird dagegen als besonders sicher angesehen.

Falls die Verschlüsselungsprotokolle mehrere Schlüssel verwenden, muss entschieden werden, welche durch QKD bereitgestellt werden sollen. Außerdem sollte beim Schlüssellebenszyklus-Management die Updatefrequenz des QKD-Keys beim Rekeying immer höher als die aktive Schlüssellebensdauer sein, um gescheiterte Versuche überbrücken zu können. Die Standards ETSI GS QKD 004/0014 liefern die allgemeine Implementierung der Schnittstelle vom Schlüsselspeicher zum Enkryptor. Dabei ist es aber nicht immer einfach möglich, eigene benötigte Schlüsselparameter zu implementieren, wie z.B. um Zeitbeschränkungen übermitteln zu können oder eine QoS einzuführen. Für Echtzeitanwendungen müssen diese Schnittstellen ausreichend schnell das Schlüsselmaterial zur Verfügung stellen.

Zudem ist die Integration abhängig von der herstellerspezifischen Implementierung der Enkryptoren. Nicht alle Switches/Router/Firewalls, die Verschlüsselung in Form von MACsec oder IPsec unterstützen, verfügen über die Möglichkeit, auch mit *Pre-Shared* Keys umzugehen und diese *Out-of-band* einzufügen. Dies ist aber essentiell für die Einspeisung von QKD-Keys in eine gewöhnliche Netzinfrastruktur. Wenn die Nutzung von Pre-Shared Keys *Out-of-band* möglich ist, dann ist diese meist kompatibel zu ETSI 014 oder auch ETSI 004. Doch meist wird eine Implementierung nach diesen Vorgaben, als zu unpassend für die reale Implementierung (z.B.: aufgrund zu weniger Parameter für Experimente oder zu sperriger Implementierung und Sicherheitsbedenken) empfunden, sodass sich mittlerweile nicht mehr alle an diesen Standards orientieren und stattdessen ihre eigenen Implementierungen einsetzen. Hierzu zählt das *Secure Key Integration Protocol (SKIP)* von Cisco oder Nokias *ANYsec*.

Die meisten höherschichtigen Sicherheitsprotokolle stellen eine Kombination aus asymmetrischen (zur Authentifizierung) und symmetrischen (zur Verschlüsselung) Kryptosystemen dar. Dabei kann bei der Authentifizierung meist ein unsicherer asymmetrischer Authentifizierungsmechanismus durch einen symmetrischen, quantenbasierten PSK ersetzt werden. Außerdem erlauben viele Protokolle von sich aus PSKs zur Verschlüsselung einzusetzen. In beiden Fällen können die benötigten QKD-Keys einfach von einem externen QKD-Gerätepaar synchron den Enkryptoren breitgestellt werden. Dies stellt allgemein die einfachste Möglichkeit dar, gängige Protokolle schnell und mit möglichst wenig Eingriffen quantensicher zu machen.

B. Quantensichere Authentifizierungsverfahren

Für das Erzielen von quantensicheren Authentifizierungsprotokollen lassen sich entweder herkömmliche Verfahren mit QKD-Keys erweitern, sollten sie noch nicht kürzlich als quantensicher weiterentwickelt worden sein, oder man bedient sich der besonderen Quanteneigenschaften auch zur Erzeugung von Authentifizierung.
Für eine sichere Integration der QKD-Keys muss zuvor eine Authentifizierung der beteiligten Partner der Schlüsselgenerierung stattfinden. Die meisten Verschlüsselungsprotokolle (A. Einsatz der QKD-Keys in verschiedenen Sicherheitsprotokolle) haben eine initiale Authentifikation der Kommunikationspartner fest integriert. Hieraus kann bei Verwendung unsicherer, nicht quantensicherer, asymmetrischer Algorithmen ein Sicherheitsproblem entstehen, da so unbemerkt eine private Schlüsselaushandlung mit einer unberechtigten Partei stattfinden kann oder diese die Nachricht modifizieren kann. Asymmetrische Methoden werden angewandt, um ein privates und öffentliches Schlüsselpaar mittels *Public Key Infrastructure* (PKI) zur Verifizierung der Kommunikationspartner im Netz zu erzeugen. Erst danach werden unabhängige weitere, nun im Fall von QKD symmetrische Schlüsselpaare erzeugt für die Verschlüsselung des Nachrichtenverkehrs.

Für eine sichere Nachrichtenkommunikation gelten neben der Vertraulichkeit (durch Verschlüsselung erreicht) folgende Grundsätze für die Authentifizierung:

- ***Authentizität*** (zweifelsfreie Identifizierung)
- ***Integrität*** (Unverfälschtheit)
- ***Verbindlichkeit*** *(Unleugbarkeit)*.

Während der eigentlichen Authentifizierung (Prüfung der Identität) eine Authentisierung (Behauptung der Identität) vorausgeht, kann erst nach einer erfolgreichen, zweifelsfreien Authentifizierung eine Autorisierung (Einräumung von Rechten) erfolgen. Eine Unleugbarkeit sollte auch im Nachhinein allen nachfragenden Instanzen gegenüber durch den Einsatz von Digitalen Signaturen gewährleistet sein.

Nachrichtenauthentizität und -integrität müssen entsprechend auch für eine quantensichere Kommunikation eingesetzt werden. Ein weit verbreiteter Ansatz, um dies zu erreichen, sind neben Message Authentication Codes (MACs) neue, als quantensicher geltende Algorithmen für Authentifizierung und Signaturen in der sog. *Post Quantum Cryptography* (PQC). Darunter gibt es sowohl *Key-Encapsulation Mechanismen* (KEM), wo ein Sitzungsschlüssel mit einem asymmetrischen Verschlüsselungsverfahren übertragen wird, als auch Digitale Signaturen, welche durch asymmetrische Signaturschlüssel Authentizität, Integrität und Verbindlichkeit bieten.

Gerade auch für QKD ist ein authentifizierter Kanal eine unbedingt notwendige Voraussetzung, um einen unbemerkten Man-In-The-Middle Angriff verhindern zu können. In der sonst gegen Mithören physikalisch sicheren Quantum Key Distribution (QKD) wird ebenfalls eine Authentifizierung für die klassischen Kanäle benötigt, welche im Post-Processing nach dem Austausch von Quantenzuständen und deren Messung in zufälligen Basen eine Kommunikation über einen sog. „Service Channel“ führen, um die vorliegenden Bits abzugleichen und daraus einen gemeinsamen Schlüssel zu erstellen. Zudem benötigt auch das KMS und die Verbindung zu den Enkryptoren zur Übertragung von QKD-Keys authentifizierte Kanäle. Eine Authentifizierung kann z.B. passwortbasiert mit extra zuvor generierten PSKs erfolgen oder mittels Zertifikaten, was eine PKI erfordert. Beim QKD-Urprotokoll BB84 [89] wird zur Authentifizierung der klassische Wegman-Carter MAC einsetzt.

Dennoch hat sich auch für nicht-quantensichere Verfahren die **Quantum Identity Authentication** entwickelt, die analog zu QKD absolut quantensicher ist. Für eine langfristige Verbindlichkeit des Nachrichteninhaltes wird die Methode der **Quantum Digital Signatures** eingesetzt. Um ebenfalls die Integrität einer Nachricht zwischen mehreren Teilnehmern zu gewährleisten, können bei der Quantenkommunikation **Quantum Secret Sharing** Verfahren eingesetzt werden, die meist auf Verschränkung oder Superposition basieren. Für die Rekonstruktion des Geheimnisses müssen dabei im Gegensatz zum **Threshold-based Secret Sharing** alle Teilinformationen bekannt sein.

Im Folgenden soll ein Überblick (siehe Abbildung 18) über verschiedene existierende klassische und quantenbasierte Ansätze, sowie neueste Entwicklungen gegeben werden, sodass der aktuelle Forschungsstand und die Entwicklung demonstriert werden.

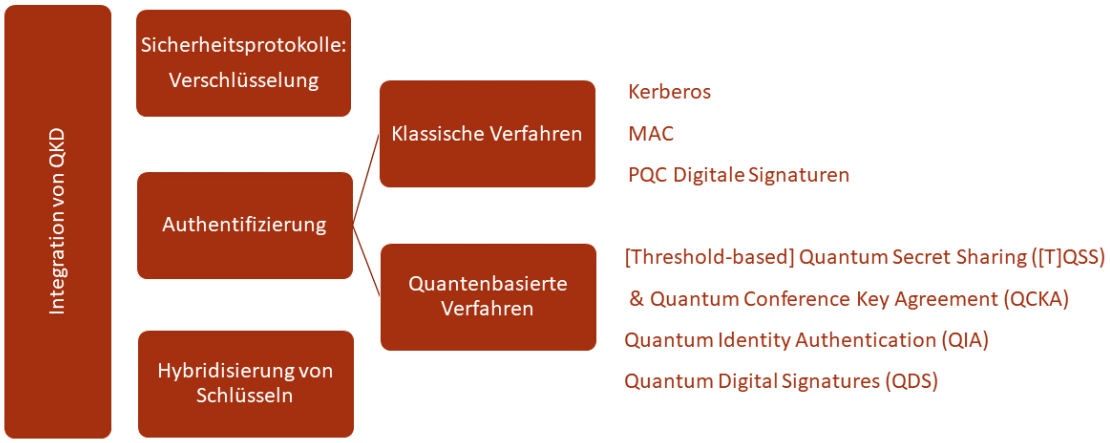


Abbildung 18: Übersicht Kapitel B: quantensichere Authentifizierungsverfahren mit der Unterteilung in Klassische Authentifizierungsverfahren, die mit QKD erweitert/ersetzt werden und hauptsächlich Quantenbasierte Verfahren, die auf der Aufbereitung/Kombination und Messung von Quantenzuständen basieren.

Nicht-Quantenbasierte Verfahren

Nicht quanten-basierte Verfahren zur Authentifizierung lassen sich meist auch mit symmetrischen QKD-Keys nutzen, um die Protokolle quantensicher zu machen. Eine einfache Methode ist die Authentifizierung basierend auf PSKs. Hier können vorab gespeicherte PSKs aus QKD stammend einfach von der Fertigung an in die Geräte integriert werden. Das Problem besteht in der Erneuerung der geheimen Schlüssel über eine gesicherte Netzwerkverbindung und in der Skalierbarkeit. Klassische Authentifizierungsverfahren sind breit etabliert: Auch bei der Authentifizierung von QKD-Protokollen wird meist ein spezieller MAC eingesetzt, allerdings in den weiteren Iterationen mit QKD-Keys. Zudem gibt es neue Entwicklungen von post-quanten Algorithmen mit verschärfter mathematischer Sicherheit, um leicht berechenbar gewordene Methoden zu ersetzen.

Kerberos und QKD

Bei Kerberos (RFC 4120 [90]) handelt es sich um ein verteiltes Authentifizierungsprotokoll mit Tickets und symmetrischer Verschlüsselung, welches häufig in unsicheren Netzwerken eingesetzt wird. Seinen Namen aus der griechischen Mythologie verdankt es der dreiteiligen Struktur bestehend aus Client,

(Applikations-)Server und Key Distribution Center (KDC). Das KDC gliedert sich in den Authentification Server (AS) und den Ticket Granting Server (TGS). Dabei fragt der Client, der initial Zugriff auf einen (Applikations-)Server wünscht, beim AS an und erhält nach Anmeldung ein Ticket, womit er sich beim Server authentifizieren kann. Wenn der Client schon ein erstes Ticket besitzt, kann er beim TGS einen Berechtigungsnachweis anfragen, um ein zweites Ticket zu erhalten. Dieses Ticket funktioniert auch als (Service) Session Key und kann (mit davon abgeleiteten Schlüsseln) für die Verschlüsselung der weiteren Kommunikation genutzt werden. Das Ticket ist dabei nicht immer effektiv verschlüsselt, sodass zusätzlich der Client sich als Authenticator ausweisen muss, was über den geheimen Session Key verschlüsselt mit einem Zeitstempel erfolgt, um Replaying zu verhindern. Dieses Protokoll ist an sich nicht quantensicher und müsste daher um QKD-Keys ergänzt werden. Die Arbeit [91] untersucht, wie Kerberos mit QKD verwendet werden könnte. Die von den KDCs pseudo-zufällig erzeugten kurzzeitigen Session Keys könnten durch echt-zufällige QKD-Keys ersetzt werden. Weitere vorgestellte Anwendungsfälle sind die regelmäßige Erneuerung von QKD-Keys für langzeitige Keys der inter-KDC-Principals mit erhöhter Sicherheit, sowie die Erzeugung des Vertrauens zwischen Cross-Realms der KDCs mittels QKD-Keys. Die KDCs der Realms eignen sich dann auch als Trusted Nodes für das QKD-Netzwerk. Da sich dieses Protokoll auf korrekte Zeitstempel stützt, sollte auch die Synchronisation mit Master/Slave über QKD-Keys abgesichert werden.

MACs: Wegman-Carter Authentifizierung bei BB84

Bei der Wegman-Carter Authentifizierung [92] beim BB84 QKD-Protokoll (Discrete Variable (DV) QKD Protokoll) handelt es sich um ein lang etabliertes symmetrisches MAC-Verfahren mit informationstheoretischem Sicherheitsbeweis, bei dem basierend auf einem Pre-Shared Key die Prüfsumme berechnet und mitgeschickt wird. Zunächst wird ein Pre-Shared Key bei der ersten Authentifizierung benötigt. Bei den weiteren Authentifizierungen wird stets ein kleiner Teil des erzeugten QKD-Schlüsselmaterials für die Generierung einer weiteren Authentifizierung mit QKD-Keys generiert: bei mehreren Nachrichten wird aus einem kleinen Zufallsschlüssel ein nachrichtenabhängiger 'Tag' (Prüfsumme) für eine beliebig große Nachricht mit einer Funktion h_k(message) erzeugt. Dabei darf die Funktion nicht aus der Nachricht mit ihrem Tag ermittelbar sein. Aufgrund der informationstheoretischen Sicherheit wird es häufig u.a. beim BB84-Protokoll (DV QKD Protokoll) und auch bei Herstellern von Geräten mit Continuous Variable QKD Protokoll wie KEEQuant eingesetzt, da es keine Schwachstelle erzeugt und die Systeme auch zukünftig komplett sicher sind.

Es besteht aber ein Problem beim Einsatz von Wegman-Carter Authentifizierung, welcher nur jeweils einen Schlüssel mit einer einzigen Nachricht im Zusammenhang mit QKD-Protokollen verschlüsseln sollte, um auch wirklich sicher zu sein: Die meisten QKD-Protokolle nutzen für die zukünftigen Authentifizierungen einen kleinen Bestandteil der vorher erzeugten QKD-Keys, sodass diese zunehmend unsicherer werden, solange bis ein neuer unabhängiger Schlüssel generiert wird [1]. Allerdings sollen die QKD-Schlüssel ja 100%-ig abhörsicher sein und somit sollte es überhaupt zu keinem Leck kommen. Darüber hinaus gibt es ein Effizienz-Problem: Durch das Benötigen von initialen Pre-Shared Keys und zusätzlich bei jeder Benutzung des authentifizierten Kanals, steigt der Realisierungsaufwand in Netzwerken exponentiell. Es ist keine einfache PKI mit Trusted Authorities möglich. Deshalb müssen geeignete Authentifizierungsmethoden für QKD-Protokolle gefunden werden. Weitere MACs mit effizienterem Schlüsselbedarf sind:

- HMAC RFC 2104 (Hash-basiert) [93]
- CMAC NIST.SP.800-38B (symm. Key Block Cipher) [94]
- GMAC NIST.SP.800-38D (Galois/Counter Mode) [95]

- Ein neu entwickelter Algorithmus in [96] zielt darauf ab, den Anteil der QKD-Keys für OTP, der für die Authentifizierung benötigt wird, in der Länge zu optimieren und in der Größe mit paarweisen verzögerten Authentifizierungen auf < 1 % zu reduzieren. Auch die Anzahl der Tags (Prüfsumme) wird auf nur einen Tag in einer Post-Processing-Runde reduziert. Zur geringeren Schlüsselverwendung durch Key-Recycling, das zu einer Offenlegung des Hash-Algorithmus nach mehreren Runden führt, wird eine Kombination von recycelten Schlüsseln für Universal_2 Polynomiales Hashing mit einem XOR Universal_2 Toeplitz Hashing und erneuerten Schlüsseln für OTP verwendet. Ein Sicherheitsbeweis wird ebenfalls geführt.

PQC Digitale Signaturen

Klassische d.h. nicht quantenbasierte Authentifizierungsmethoden für QKD bieten den Vorteil, dass keine Quantenkanäle geschaffen werden müssen und sie häufig langjährig erforscht sind. Dennoch muss auch hier auf die Quanten-Sicherheit der Algorithmen geachtet werden. Deshalb gibt es in diesem Bereich ebenfalls neue Entwicklungen, besonders im Bereich von PQC-Algorithmen (Lattice/Hash-based Digital Signatures), die erst 2024 von der NIST standardisiert wurden und somit im Moment als sicher angesehen werden können. PQC-Algorithmen, die wie bisherige RSA/ECDH funktionieren, basieren auf anderen (mathematischen) Methoden, und gelten als quantensicher (siehe auch PQC-Kapitel [97]). Dennoch gibt es im Gegensatz zu QKD keine 100 % Sicherheitsgarantie auf Quantensicherheit und manchmal kommt es zu Performance Problemen durch die aufwändigen Berechnungen. Dafür werden keine Pre-Shared Keys oder Hardware in Form von Quantenkanälen benötigt.

Zu den standardisierten Algorithmen als Federal Information Processing Standards (FIPS) vom NIST im August 2024 zählen:

- FIPS 203
  Module-Lattice-Based Key-Encapsulation Mechanismus (ML-KEM) Standard basierend auf **CRYSTALS** (Cryptographic Suite for Algebraic Lattices)**-KYBER** [98]
- FIPS 204
  Module-Lattice-Based Digital Signature Standard (ML-DSA) basierend auf **CRYSTALS-Dilithium**; basiert auf dem Short Integer Solution Problem [99]
- FIPS 205
  Stateless Hash-Based Digital Signature Standard (SLH-DSA) basierend auf **SPHINCS+** (Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures) [100]

Im Entwurf sind derzeit folgende Standards:

- FIPS 206 (Standard im Entwurf)
  FALCON (FAst fourier Lattice-based COmpact signatures over NTRU) für Digital Signatures; basiert auf dem Short Integer Solution Problem bei NTRU (Open-Source Public-Key Cryptosystem) Gittern
- HQC Hamming Quasi-Cyclic (neu geplanter Backup-KEM-Standard [101])
  Error-Correcting Code mit Generatormatrix auf einem Vektorraum mit Hamminggewicht ω; IND-CCA2-secure (Ununterscheidbarkeit von Geheimtexten) [102]

Während es sich bei CRYSTALS-Kyber und HQC um Key Encapsulation Mechanism (KEM) handelt, so stellen CRYSTALS-Dilithium, SPHINCS+ und Falcon Algorithmen für Digitale Signaturen dar und eignen sich auch für Authentifizierung. Crystals-Kyber & -Dilithium sind gitterbasierte Verfahren, deren mathematisches Problem in der Berechnung des nähesten Kreuzungspunktes des Gitters zum

Nullpunkt bzw. zu einem der Nachricht nähesten Punkt des Gitters besteht. SPHINCS zählt zu den hashbasierten Signaturen und ist zustandslos.

Beispiele für die Anwendung dieser erst kürzlich standardisierten Algorithmen findet man u.a. in einer Implementierung von Dilithium 2 & Falcon 512 im QUIC Protocol (UDP & TCP/TLS) für post-quanten Authentifizierung, welche einen schnelleren Handshake wie RSA bietet [103]. Eine Authentifizierung für QKD mittels der neu standardisierten PQC-Algorithmen ist noch nicht viel erforscht, aber in [104] werden die klassischen Kanäle der Authentifizierung von QKD mit PQC CRYSTALS-Kyber, -Dilithium und Falcon abgesichert. [105] verwendet für die Authentifizierung PSKs aus QKD und die kürzlich ausgewählten NIST PQC-Signaturschemata (Dilithium, Falcon, SPHINCS+), sowie das zustandbehaftete hashbasierte PQC eXtended Merkle Signatur Schema (XMSS). Während PSKs auf der simulierten Strecke deutlich schnellere Authentifizierung ermöglichen, bringt Dilithium trotz seiner größeren Schlüssellänge Geschwindigkeitsvorteile gegenüber Falcon mit.

Zudem eignen sich für eine quantensichere Authentifizierung eher PQC Digitale Signaturen, wie in [106] demonstriert wurde. Für den hybriden Einsatz von QKD und PQC zu Authentifizierungszwecken siehe Kapitel „Hybride Authentifizierung“.

Quantenbasierte Verfahren

Da QKD sich nur auf die quantentechnische Umsetzung für Vertraulichkeit, also Verschlüsselung konzentiert, gibt es analoge Forschungsbewegungen bei der Authentizität.

Um quantensichere Authentifizierung in Zukunft für klassische Algorithmen, in die sich keine QKD-Keys als PSK zur Authentifizierung sicher integrieren lassen, zu ermöglichen, werden neue eigenständige quantenbasierte Methoden zur Authentifizierung entwickelt: Bei diesen Methoden wären allerdings Quantenspeicher bzw. Quantenrepeater nötig, um verschränkte Zustände über längere Zeit zu halten. Um bereits ab der Verfügbarkeit von langandauernden Quantenspeichern, quantensichere Authentifizierungstechniken zur Verfügung zu haben, die rein auf physikalischen Quanteneigenschaften basieren, ist die Erforschung dieser Methoden auch jetzt schon wichtig.

Bislang ist die Authentifizierung zwischen Gesprächspartnern verglichen zu der unbedingt erforderlichen Langzeitsicherheit bei den Schlüsseln zur Verschlüsselung noch als weniger kritisch zu sehen, da die kurzzeitige Sitzung meist beendet ist oder regelmäßig neu authentifiziert worden ist, bevor die Authentifizierung von Quantencomputern berechnet werden konnte. Doch in Zukunft könnten gerade langandauernde authentifizierte Sitzungen, die nicht ständig neu authentifiziert werden können, anfällig gegenüber schnell-rechnenden Quantencomputern werden und so könnten die Authentifizierungsinformationen noch innerhalb der bestehenden Session von anderen ausgenutzt werden. Auch für längergültige X.509 Zertifikate gibt es aktuell noch keine quantensichere Technologie; hier bieten sich quantenbasierte Digitale Signaturen an.

Im Folgenden werden verschiedene quantenbasierte Authentifizierungsverfahren in ihren Grundzügen dargestellt:

- *Quantum Secret Sharing* (QSS)
- *Threshold-based QSS* (TQSS)
- *Quantum Conference Key Agreement* (QCKA)
- *Quantum Identity Authentication* (QIA)

- *Quantum Digital Signatures* (QDS)

Dazu werden als Beispiele einige potentiell umsetzbare Verfahren ohne Quantenspeicher aus der aktuellen Forschung herausgegriffen.

Quantum Secret Sharing (QSS), Threshold-based (TQSS) & Quantum Conference Key Agreement (QCKA)

Quantum Secret Sharing (QSS)

QSS dient für viele nachfolgende Quantenauthentifizierungsverfahren als Grundlage. Es findet somit Anwendung in der QDS, sowie teilweise bei QIA und ähnelt QCKA: Zu den Grundsätzen von QSS zählen Zuverlässigkeit, Vertraulichkeit und die Fähigkeit zum Austausch von Quanteninformationen zwischen Teilnehmern. Dabei wird Information unter zwei oder mehr (n) Parteien aufgeteilt, sodass nur alle (n) zusammen die Original-Information wieder herstellen können. Dies bezeichnet man auch als (n,n) Schwellwertschema. Bob kann z.B. die Nachricht von Alice nicht alleine entschlüsseln, solange er nicht mit Charlie und den übrigen Parteien zusammenarbeitet. Dabei kann es sich um einen Quantenzustand oder um einen Schlüssel handeln. Generell kommt dieses Verfahren zum Einsatz, wenn die Gefahr besteht, dass eine Partei unehrlich ist. Es wird darauf gebaut, dass die ehrlichen Parteien die unehrlichen daran hindern, Schaden anzurichten und somit die Gruppe an sich vertrauenswürdig ist.

QSS wurde um die Jahrtausendwende in vielen theoretischen Arbeiten erforscht und wird es immer noch, allerdings gibt es wenige experimentelle Arbeiten und die Algorithmen sind, wie die übrigen Verfahren, noch weit von einer praktischen Realisierung entfernt, was sich allerdings mit Fortschritten in der Teleportation oder dem Entanglement-Swapping ändert.

QSS wird in [107] grundsätzlich in die QSSCM (Quantum Secret Sharing Classical Messages) und die SSQI (Secret Sharing Quantum Information) unterteilt. Während bei QSSCM, einem (n,n)-Schwellwertschema, das nur einzelne Photonen verwendet, das Geheimnis eine beliebige Nachricht sein kann, kann es bei SSQI ein beliebiger unbekannter Quantenzustand sein. SSQI basiert auf Teleportation und ist damit irrelevant für in naher Zukunft praktisch realisierbare Verfahren. Bei QSSCM mit drei Teilnehmern präpariert Bob einzelne polarisierte Photonen, während Charlie zufällig aus drei unitären Operationen wählt, diese auf die von Bob erhaltenen Photonen anwendet und an Alice sendet. Alice bittet zufällig Bob oder Charlie zuerst um seine Information und führt die gleichen unitären Operationen wie Charlie aus. Durch anschließendes Messen mit Bobs Information bestimmt sie die Fehlerrate. Anschließend encodiert Alice ihr Photon je nach Informationsbit mittels unitärer Operation. Die encodierten Photonen schickt sie an Charlie, sodass dieser mit Bob zusammenarbeiten muss, um Alices Geheimnis zu erhalten; Alice veröffentlicht wie bei allen Quantenaustauschverfahren Bits, um sicherzustellen, dass es keinen Lauscher gab.

[108] zeigt, wie QSS nicht nur zum Informationsaustausch, sondern auch für eine Authentifizierung mit mehreren Partnern (Bobs) durch verschränkte d-level mehrteilige Greenberger-Horne–Zeilinger (GHZ)-Zustände [109] realisiert wird. Mit den im Vorfeld von Alice eingefügten Detektions-Partikeln eines GHZ-Zustandes werden zwei Detektionssequenzen generiert: die eine Sequenz D_i^2 einspricht einer Hälfte eines zufälligen privaten Strings Q und die andere Sequenz D_i^1 einer Hälfte eines privaten One-Time Keys K. Diese K-Schlüsselsequenzen dienen den unterschiedlichen Bobs zur Authentifizierung von Alice, während die Q-Sequenz von Alice dazu dient, verschiedene Bobs zu authentifizieren. Zur Authentifizierung von den mehrfachen Bobs gibt Alice die Position der Detektions-Partikel der zweiten Detektionssequenz für jeden Bob an. Jeder Bob führt dann eine Messung an der zweiten Detektionssequenz durch und teilt Alice das Ergebnis mit. Bei Unverfälschtheit

stimmt die Messung mit dem vorbestimmten Wert $t_{i,j}$ (zweiten Teil von Q) überein (siehe Abbildung 19). Umgekehrt kann die Identität von Alice geprüft werden, indem Alice die Position und Basis der Detektions-Partikel der ersten Detektionssequenz für jeden Bob festlegt, welche von den Bobs gemessen wird. Dieses Messergebnis sollte mit dem vorbestimmten Wert $k_{i,j}$ (erster Teil von K) übereinstimmen. Dieses Schema lässt sich zudem erweitern, um auch Secret Sharing und Recovery mit zusätzlichen Receivern als User und einem Restorer durchzuführen.

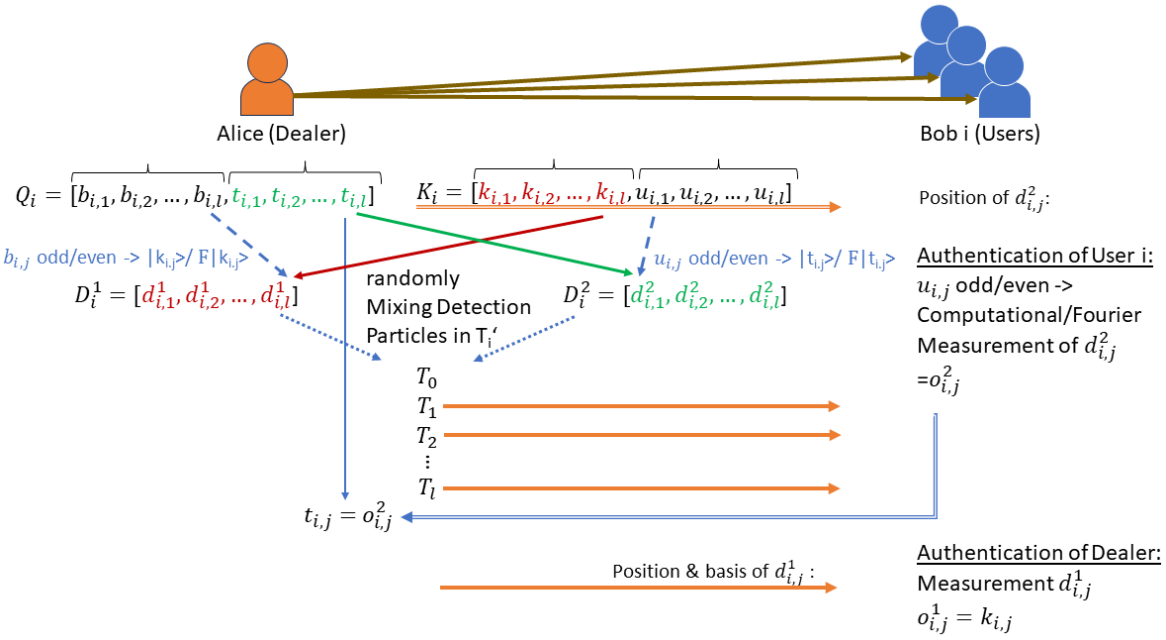


Abbildung 19: ***Authentifizierung mit QSS*** *[108]: Initiales Erzeugen eines String Q und eines String K für jeden Bob durch Alice; Verteilung K_i an jeden Bob;* ***Authentifizierung eines Bob Users****: empfangene Position der Detektionspartikel der zweiten Detektionssequenz wird von Bob abhängig vom Wert u_i,j gemessen; Ergebnis wird an Alice zur Abgleichung mit t_i,j (zweite Hälfte String Q) gesendet;* ***Authentifizierung Alice Dealer****:gesendete Position & Basis erster Detektionssequenz wird von Bob gemessen und mit k_i,j (erste Hälfte String K) abgeglichen.*

Threshold-based QSS (TQSS)

Das Besondere bei *Threshold-based QSS* (TQSS) ist, dass die Information auch ohne die Kenntnis aller (n) Teile rekonstruiert werden kann, solange mindestens (k) oder mehr Teilstücke vorliegen. Hier spricht man von einem (k,n) Schwellwertschema. Dagegen offenbaren weniger als k Teilstücke nichts von der Geheiminformation. So muss nicht zwangsläufig von allen Teilnehmern ein Stück des Geheimnisses erhalten werden, um in Besitz des vollen Geheimnisses zu kommen, aber dennoch von ausreichend vielen Parteien entsprechend dem vorher festgelegten Schwellwert –, wodurch auch von einer Ehrlichkeit der verbleibenden Parteien ausgegangen werden kann.

Ein Beispiel dafür ist ein (2,n)-threshold LOOC-QSS Schema mit lokalen Quantenoperationen und klassischer Kommunikation [110], bei dem jede zwei Teilnehmer das Geheimnis immer auf der Grundlage der lokalen Unterscheidbarkeit der orthogonalen n-Qu**D**it-GHZ-Zustände rekonstruieren können. Dabei werden die Eigenzustände der übrigen Teilnehmer als Messbasis genutzt, um die Projektion fortzuführen und das Ergebnis im Produkt der Eigenwerte zu erhalten. Ein weiteres Beispiel ist der (3,5)-TQSS mit maximal sechs verschränkten Qu**B**it-Zuständen, bei dem zwei Teilnehmer (nach Mitteilung des Messergebnis von einem beliebigen Bob) eine Bell-Messung durchführen und der Dritte eine lokale Messung auf der Basis des ersten Bobs, um das Geheimnis zu erhalten [111].

Quantum Conference Key Agreement (QCKA)

QCKA stellt eine Art Multi-User QKD dar und hat darüber hinaus Ähnlichkeit zu QSS, bei dem der Schlüssel das gemeinsame Geheimnis der Teilnehmer darstellt, die Teilnehmer jedoch nicht zusammenarbeiten müssen. In QCKA Protokollen tragen alle Teilnehmer gleichsam zur Erzeugung und Verteilung der einzelnen Schlüssel bei. Nach der Übereinstimmung, erhalten alle Parteien dieselben erzeugten Schlüssel. Somit bietet sich dieses Verfahren besonders für die Kommunikation in Multi-User Netzwerken an.

Häufig wird dies über die Erzeugung von verschränkten Zuständen (meist GHZ) realisiert. Es gibt weitere verschiedene Arten mehrteilige Verschränkungen zu erzeugen, wie mit dem W-Zustand [112] oder mit bestimmten mehrteiligen Bell'schen Ungleichungen [113].

Der Zweck von QCKA besteht darin, einen gemeinsamen geheimen Schlüssel unter n Teilnehmern zu etablieren. Alle Teilnehmer können die öffentlichen Nachrichten verschlüsseln und die verschlüsselten öffentlichen Nachrichten entschlüsseln, die von anderen Teilnehmern gesendet werden. Dabei können die Lauscher keine öffentlichen Nachrichten entschlüsseln, die von den Teilnehmern gesendet werden. [114] zeigt einen experimentellen Hardwareaufbau, der sowohl für QSS als auch für QCKA geeignet ist, wobei ein Wechsel zwischen den Protokollen nur in Software erfolgt. Mit dem von Alice für alle Bobs generierten Quantenschlüsseln kann der von Alice stammende gemeinsame geheime Schlüssel von allen individuell decodiert werden.

Quantum Identity Authentication (QIA)

QIA dient zur zweifelsfreien Identifikation eines Kommunikationspartners durch das Ausnutzen von Quantenzuständen. Meist wird eine Aufgabe an den Partner gesendet, der diese nur aufgrund seines Wissens korrekt bearbeiten kann und sich so durch Rücksenden der korrekten Antwort zweifelsfrei gegenüber dem anfordernden Partner identifizieren kann. Dies ist auch bekannt als Challenge-Response Mechanismus. Im Unterschied zu klassischen Authentifizierungsschemata wird hierzu ein Quantenzustand gesendet. Zudem besteht bei der quantenbasierten Authentifizierung der Vorteil, dass ein Abhören/Messung analog zu QKD bemerkt werden würde. Es ist zu beachten, dass viele Ansätze in der Literatur auf Quantenspeichern (z.B. Direct Communication oder Third Party Protokolle) beruhen und somit bisher nicht realisierbar sind. Auch Ansätze wie Teleportation (wegen Rauschen im Kanal) oder Verschränkung (Erhaltung der Zustände) sind sehr schwierig umzusetzen.

QIA gliedert sich in mehrere Bereiche, wie Abbildung 20 zeigt [115], [116]: während die verschränkungsbasierten Protokolle, welche meist auf Quantenspeichern beruhen, hier nicht weiter betrachtet werden, gibt es auch eine Klasse mit nicht-verschränkungsbasierten Protokollen, wobei diese z.B. auf der Superposition oder orthogonalen Produkten der Zustände beruhen. Zu der Klassifikation anhand von quantenbasierten Kommunikationsaufgaben zählen z.B. Protokolle basierend auf QSS [117] und auf QKD mit Pre-Shared Keys. Im Folgenden sollen zu Demonstrationszwecken einige nicht-verschränkungsbasierte QIA-Protokolle erläutert werden. Zudem können sich aus speziellen Quantenberechnungen weitere Methoden zur QIA ableiten: z.B. der Quantum Error Detection/ Correction Code, bei dem es gilt einen Kohärenzzustand mit rauschfreien Unterraum zu finden [118] und die Quantum Private Comparison, bei der zwei die Gleichheit ihrer Geheimnisse mit Hilfe eines halbehrlichen Dritten über Quanten vergleichen können, ohne dass es diesem offen liegt [119], [120].

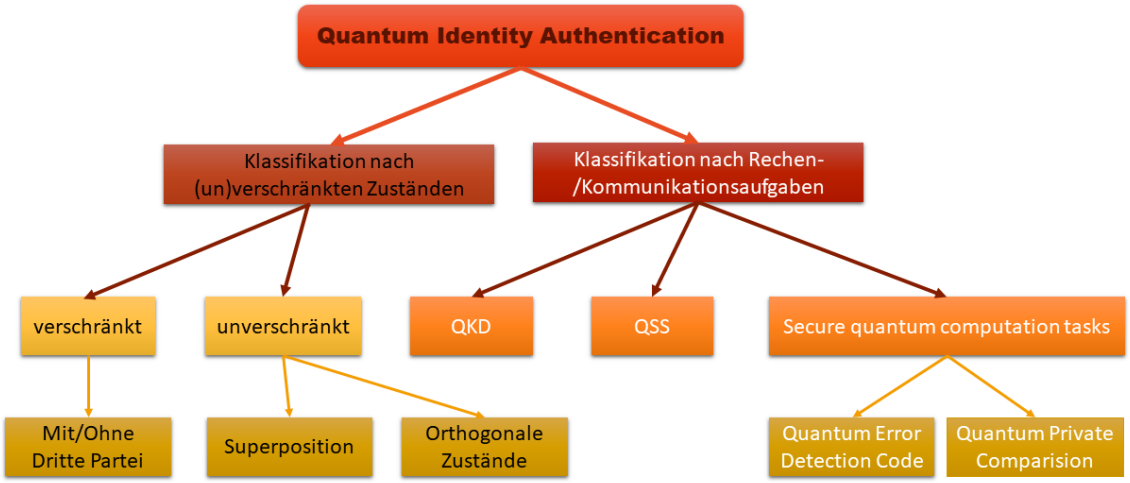


Abbildung 20: Klassifikation von QIA Methoden (gekürzt nach [115], [116]): Klassifkation nach Unterscheidung von verschränkten Zuständen oder nicht, und nach den Aufgaben zur Authentifizierung an sich.

In einem realitätsnahen QIA-Protokoll können Alice und Bob mit einem einzigen Photon zwei Bits einer Authentifizierungssequenz authentifizieren [121]: Zur Kodierung wählt Alice den Authentifizierungs- oder den Sicherheitsmodus (ähnlich zum BB84-Protokoll) aus: Im Authentifizierungsmodus erzeugt Alice ein Authentifizierungs-QubBit gemäß dem Authentifizierungsschlüssel A_k und sendet einen schwachen kohärenten Zustand pro Slot mit abgeschwächter Intensität an Bob. Im Sicherheitsmodus bereitet Alice den Decoy State vor, um die Sicherheit des Quantenkanals zu überprüfen. Nach der Übertragung misst Bob den Decoy-State mit Basis gemäß dem Authentifizierungsschlüssel A_k. Anschließend kündigt Alice öffentlich den Modus und den Zustand des Decoy States an. Damit vergleicht Bob sein Messergebnis: wenn Übereinstimmung, dann gab es keinen Angreifer oder Systemfehler. Bei der Dekodierung misst Bob das von Alice generierte Authentifizierungs-QuBit mit dem Authentifizierungsschlüssel als Basis. Dies kann mit einem Phasenmodulator, einem Strahlteiler und Detektoren durchgeführt werden. Im anschließenden Post-Processing führen Alice und Bob Fehlerkorrektur durch und ermitteln die QBER. Dabei bestimmt Bob, ob die Authentifizierung bestanden wird oder nicht. Hier handelt es sich um einen praktischen Vorschlag, aber die Abhörsicherheit ist immer noch nicht eindeutig geklärt.

Das theoretische QIA-Protokoll [122] mit nicht-orthogonalen Zuständen mit vertrauenswürdigem Dritten (Alice) und zwei zusätzlichen Teilnehmern (Bob, Charlie) basiert auf keiner Verschränkung. Es ist kompatibel mit QKD und erfordert vorab Pre-Shared Secrets zwischen Benutzern und dem Dritten. Während der Vorbereitung werden aus den Pre-Shared Keys drei-Bit-Positionen von den anderen zwei Partnern zufällig ausgewählt, um entsprechend zu zwei drei-Bit-Sequenzen kodiert zu werden. Anhand der Bitpositionen encodieren Bob und Charlie mit daraus abgeleiteten Basen diese als Qubits. Alice prüft auf Richtigkeit und erzeugt bei Erfolg aus Teilen der beiden Pre-Shared Keys den Schlüssel mit bitweisem XOR, sodass sich Stellen mit identischen Bitpositionen ergeben. Diese werden dann für die gegenseitige Authentifizierung zwischen Bob und Charlie ausgenutzt, die ebenfalls auf dem gegenseitig überprüfenden Encodieren von drei Bits besteht. Die Authentifizierungsschlüssel werden nur einmal verwendet. Die Methode soll resistent gegen Man-in-the-Middle-Angriffe sein, obwohl der klassische Kanal, der für den initialen Request und die Bitpositionen genutzt wird, nicht authentifiziert ist. Die Wahrscheinlichkeit, dass Eve fälschlicherweise die Photonen richtig encodiert, soll durch eine hohe Anzahl an zu überprüfenden Photonen kompensiert werden. Dennoch beträgt die Wahrscheinlichkeit einer Authentifizierung von Eve noch 1 % bei 16 überprüften Photonen. Das theoretische Modell weist

zudem auf die Herausforderungen einer praktischen Implementierung, wie Einzelphotonen-Detektion und den Umgang mit Kanalverlusten und Fehlerraten hin.

Das QIA-Protokoll ohne vertrauenswürdigen Dritten unter Verwendung ternärer homomorpher Quantenverschlüsselung (QHE = beliebige Quantentransformation auf verschlüsselten Daten ohne Datenentschlüsselung) [123] verwendet ternäre Qubit-Rotationen mit einem ternären einzeln unitären Operator, der mit einer Gruppe von ternären einzelnen Quanten-Elementargattern in Quanten-Schaltkreisen ausgewertet werden kann. Ein Qutrit ist für die Verifikation von zwei Bits erforderlich, sodass es Quantenressourcen spart. Unter der Annahme, dass A und B im Voraus eine gemeinsame geheime Matrix haben, werden Quanten verwendet, um zu überprüfen, wie gut beide Parteien die Zeichenfolge kennen. So bereitet Alice die Matrix mit Check-Code vor und schickt sie an Bob als Authentifizierungsanfrage. Bob wendet eine Rotation darauf an und sendet sie zurück an Alice, die Rotation mit dem Check-Code anwendet und den Quantenzustand bitweise mit den drei Basen misst. Stimmt das Messergebnis mit dem Set der ursprünglichen Matrix überein, so wird die Authentifizierung erfolgreich bestanden.

Neben der QIA, die die bloße Identität der anfragenden Partei durch Besitz geheimer Informationen legitimiert, gibt es zusätzlich noch die Q*uantum Message Authentication* (QMA), die den Ursprung und die Integrität einer Nachricht sicherstellt. Diese Form unterscheidet sich nur durch die fehlende Verbindlichkeit/Nicht-Abstreitbarkeit (non-repudiation) von den *Quantum Digital Signatures* (QDS) [124].

Quantum Digital Signatures (QDS)

Eine Digitale Signatur (asymmetrisches Verfahren) wird mittels Quantenzustand über QKD-Kanäle erzeugt, sodass der Empfänger die Identität des Senders und die Integrität der Nachricht auch im Nachhinein noch zweifelfrei identifizieren kann. Dabei findet die Verifizierung meist über klassische Kanäle statt. Außerdem kann der Sender bei einer Digitalen Signatur nicht abstreiten, dass er eine Nachricht gesendet hat (non-repudiation).

Der Unterschied zwischen Authentifizierung und Digitaler Signatur ist, dass eine Authentifizierung die bloße Zertifizierung der Identität des Nachrichtenerstellers durch Kenntnis des gemeinsamen Schlüssels für sich beweist, während die Digitale Signatur auch einer dritten Partei zu einem späteren Zeitpunkt ermöglicht mit dem Public Key zu verifizieren, dass die gesendete Nachricht von dem behaupteten Absender stammt und nicht durch Parteien mit Kenntnis des symmetrischen Schlüssels modifiziert wurde.

Allerdings beruhten bisher die meisten Protokollvorschläge auf zur Verfügung stehenden Quantenspeichern, da hierfür meist ein Abspeichern der Quantenzustände erforderlich ist. Erst neuere Protokolle konzentrieren sich auf QDS ohne Quantenspeicher:

Eingeführt wurden QDSs ohne Quantenspeicher z.B. mit [125], welches diese umging, indem in der Verteilungsphase die Quanten-Signaturen direkt gemessen werden und als klassische Informationen bei den Empfängern vorliegen, während die Verifizierung klassisch mit authentifizierten Kanälen (z.B. Vergleich zweier Zustände; QKD-Pre-Shared Keys) erfolgt. Dabei liegt die dennoch informations-theoretisch sichere Umsetzung am jeweiligen Protokoll.

In [126] wird beispielsweise ein praktischer Austausch einer Digitalen Signatur in einem skalierbaren Quantennetzwerk mit Measurement-Device-Independent (MDI)-Protokoll gezeigt. Dabei wird ohne

direkte Verbindung von Alice zu Bob über je 25 km Faser eine Verbindung von Alice-Charlie oder Charlie-Bob (QKD) hergestellt und mit beiden nicht-blockierenden Intensitätsmodulatoren Alice-Charlie-Bob (MDI) realisiert (siehe Abbildung 21). Alice stellt im vorgestellten Fall den Signierer dar, Charlie den Verifizierer und Bob den Empfänger. Eine MDI-QKD ohne Error Correction and Privacy Amplification wird für das Senden einer Signatur von Alice zu Bob verwendet und QKD ohne Error Correction and Privacy Amplification, um die für die Signatur zwischen Alice zu Charlie zu transportieren. Für eine Symmetrisierung wird auch zu Schlüsselverteilung reines QKD zwischen Bob und Charlie eingesetzt. Dabei werden alle Bits der X-Basis und zusätzlich ein paar Bits der Z-Basis für die QBER-Ermittlung offengelegt, während der Rest für die Erzeugung der Digitalen Signaturen dient. Das Protokoll basiert auf der ansammelnden Generierung von Datenblöcken, solange bis möglichst wenig statistische Abweichungen auftreten. Aus jedem Block können mehrere Signaturen erzeugt werden. QDS erreicht je nach Blocklänge eine Rate von einem vorzeichenbehafteten Bit pro 45 s für QDS mit MDI-QKD bzw. 72 ms für QDS mit QKD.

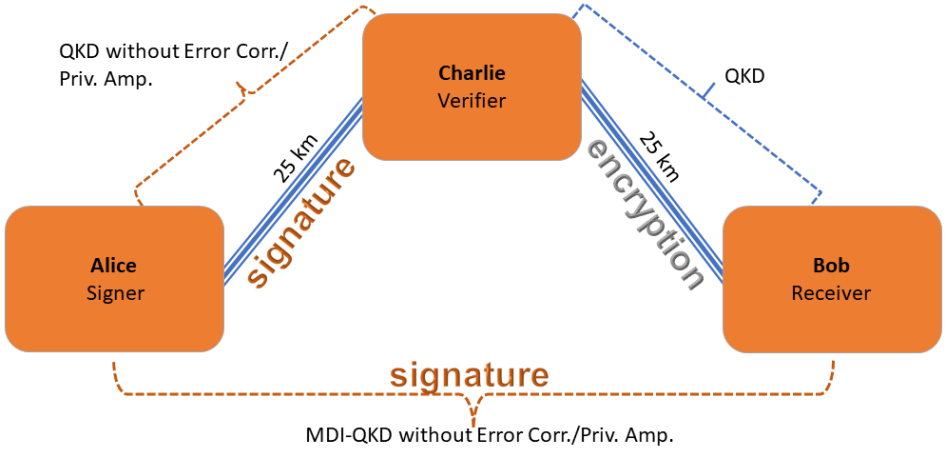


Abbildung 21: Generierung Signatur zwischen Alice (Signer)-Bob (Receiver) und Alice-Charlie (Verifier) mit MDI-QKD und QKD jeweils ohne Error Correction und Privacy Amplification, normale QKD wird für die Strecke Charlie-Bob eingesetzt [126].

Ein anderes auf Effizienz ausgelegtes QDS-Protokoll unter Verwendung von asymmetrischen Quantenschlüssel durch Secret Sharing, Conference Key Agreement, Einmalverschlüsselung (OTP) und von One-time universal$_2$ Hashing (OTUH), wird in [127] vorgestellt. So soll ein 348 Bit-Schlüssel ein Dokument der Länge 2^{64} signieren können. Auch hier werden vorgenerierte Schlüsselsequenzen X_a und Y_a von QKD oder QSS zur Vorbereitung der Signaturen verwendet. Alice (Signierer) sendet diese an Bob (Charlie) X_b Y_b und Charlie (Bob) X_c Y_c, welcher der Verifizierer wird (siehe Abbildung 22). Es werden sechs Schritte zur Signierung von Alice angewandt:

1. mit einer zuerst generierten Zufallszahl wird ein irreduzibles Polynom generiert;
2. damit wird zusammen mit der initial vorbereiteten Schlüsselsequenz X_a eine Linear Feedback Shift Register (LFSR)-basierte Toeplitz Matrix generiert;
3. eine Hashoperation wird damit auf dem Dokument ausgeführt;
4. ein Hashwert wird mit dem irreduziblen Polynom ermittelt;
5. dieser Hashwert wird mit der initialen Schlüsselsequenz Y_a OTP verschlüsselt;
6. Hashwert wird über den öffentlichen Kanal an Bob gesendet.

Zur Authentifizierung werden über den klassischen Kanal die von Alice erhaltene Signatur mit dem Dokument und die initialen Schlüsselsequenzen an Charlie übertragen, ebenso diese von Charlie zu

Bob. Mit XOR werden jeweils neue Schlüsselsequenzen erhalten mit Schlüsseln von Bob und Charlie. Bob erhält nach Zusammenarbeit mit Charlie mit weiterem XOR den erwarteten Hashwert und das irreduzible Polynom. Bob kann damit eine Toeplitz Matrix und daraus einen Hashwert generieren, den er zur Verifizierung abgleicht. Anschließend führt Charlie dasselbe Prozedere durch zur Verifizierung.

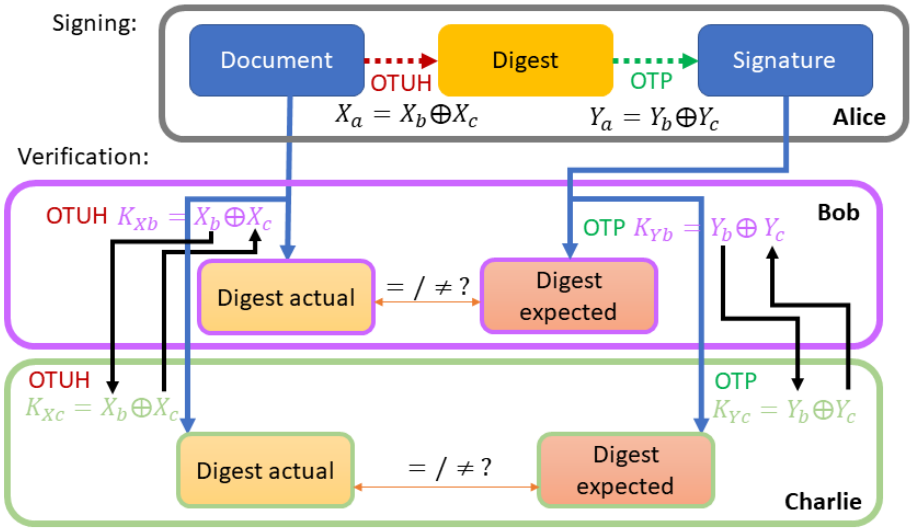


Abbildung 22: Effizientes QDS-Protokoll mit Einmalverschlüsselung (OTP) und One-time universal$_2$ Hashing (OTUH): Alice (Signierer) sendet vorgenerierte Schlüsselsequenzen X_a und Y_a von QKD oder QSS an Bob (Charlie) X_b Y_b und Charlie (Bob) X_c Y_c, welcher der Verifizierer wird. Bob (Charlie) erhält nach Austausch mit Charlie X_c Y_c (Bob X_b Y_b) mit weiterem XOR K_{Yb}(K_{Yc}). Bob(Charlie) generiert mit dem daraus abgeleiteten irreduziblen Polynom und mit K_{Xb}(K_{Xc}) eine Toeplitz Matrix und daraus einen Hashwert, den er mit der Actual & Expected Digest abgleicht. [127]

Fazit zu Authentifizierungsmethoden mit QKD

Bei QKD ist das Bestehen einer sicheren Authentifizierung nur vor und während des Protokolls notwendig, da die Sicherheit der erzeugten QKD-Schlüssel unabhängig vom Fortbestehen der Authentifizierung ist. Die Authentifizierung muss nach Beenden des Protokolls nicht mehr zwingend sicher bleiben [128]. Auch bei Erneuerung der PSKs müssen die alten PSKs nicht zwingend geheim bleiben, da es sich im Fall von QKD um echt zufällige Schlüssel handelt, aus denen sich keine Schlüsse für weitere oder daraus abgeleitete Schlüssel ergeben. Somit können auch klassische Verfahren zur Authentifizierung bei QKD-Protokollen verwendet werden, solange sie zum derzeitigen Zeitpunkt sicher sind. Anders sieht es bei Protokollen aus, die keine QKD-Keys verwenden bzw. nicht quanten-sicher sind: diese sind auf eine absolut sichere Authentifizierung und Integrität im Zeitalter der Quantencomputer angewiesen, um Man-In-The-Middle-Angriffe zu verhindern.

Quantenbasierte Methoden zur Authentifizierung sind allerdings sehr theoretisch und derzeitig kompliziert in der Realisierung bzw. noch unmöglich, wenn die Verfahren auf kurzzeitigen Quantenspeichern beruhen. In diesen mit der Quantenkommunikation verwandten Gebieten wird seit vielen Jahren kontinuierlich geforscht. Viele Verfahren basieren auf noch nicht kommerzialisierten Technologien, wie „Entanglement Swapping“ oder „Teleportation“ und können nicht über weite Distanzen realisiert werden. Zudem ist für eine praktische Umsetzung im Netz die Skalierbarkeit und die Rate der Verfahren wichtig. Deshalb ist eine weitere praktische Realisierung dieser theoretischen quantenbasierten Authentifizierungsprotokolle von größter Wichtigkeit. Hinzu kommt, dass es kaum etablierte Verfahren gibt, dafür eine Vielzahl unterschiedlichster theoretischer Protokollvorschläge, damit die meist nur unter Laborbedingungen umsetzbaren Protokolle, auch in der Praxis umsetzbar werden. Die wenigsten Forschungsarbeiten verfügen über einen Sicherheitsbeweis, der natürlich essentiell ist für eine sichere Realisierung. Somit ist die Entwicklung und Standardisierung dieser prinzipiell sehr vielversprechenden Verfahren noch in weiter Ferne, trotz des Potentials.

Gerade der Einsatz von TQSS für die QCKA bietet sich in Zukunft als sichere Alternative zur Ermöglichung von Multi-User Netzwerken an oder von High-Dimensional QKD Protokollen, die sich ebenfalls noch in der Entwicklung befinden.

Die auf (Quantum) Digitalen Signaturen basierenden Authentifizierungsmethoden bieten den Vorteil, dass sie mittlerweile (recht) recheneffizient sind und somit die Performance des Systems nicht beeinflussen und auch nachträglich langzeitig Nicht-Abstreitbarkeit des Nachrichteninhaltes bieten. Deshalb sind QDS derzeit das vielversprechendste Forschungsgebiet von quantenbasierten Authentifizierungsverfahren.

Für die Authentifizierung und Digitale Signaturen sollte aber bis auf Weiteres auf klassische bzw. auf hybride Verfahren gesetzt werden, nicht zuletzt wegen der fehlenden Verfügbarkeit von langzeitigen Quantenspeichern. Die Verfügbarkeit würde eine Umsetzung der bis dahin gut erforschten Authentifizierungsmethoden gänzlich quantenbasiert und -sicher (ohne klassischen Kanal) ermöglichen. Weiteres zu Vorteilen von Hybridisierung von quanten-basierten mit klassischen Verfahren für Authentifizierung und Signaturen findet sich im Kapitel „Hybride Authentifizierung“.

C. Hybridisierung mit QKD

„Hybrid“ bezieht sich normalerweise auf den Schlüsselaustausch, der mehrere kryptographische Methoden zum Schlüsselaustausch vereint. Darunter können mehrere der folgenden Methoden miteinander kombiniert werden: QKD, PQC, DH, PSK; Dies dient dazu, um die Sicherheit der einzelnen Komponenten zu stärken. Falls die Schlüssel einer Methode offengelegt werden, so soll durch die andere(n) Methode(n) eine komplette Offenlegung bzw. Ausfall verhindert werden, da ein Angreifer alle Methoden brechen und die exakte Key Derivation Function (KDF) kennen müsste. Somit soll erreicht werden, dass ein Schlüssel mindestens genauso sicher ist, wie jede einzelne dieser Methoden. Ein weiterer Vorteil besteht in der dadurch vorhandenen Ausfallsicherheit bei Denial of Service Attacken (DoS), die eine Methode blockieren und so einen Mangel an Schlüsselmaterials provozieren.

Dabei ist zu beachten, dass es noch keine Standard-Definition von “hybrid” in der Kryptographie gibt. Zudem ist es wichtig, hybriden Schlüsselaustausch von hybrider Verschlüsselung mit Public Key Infrastructure (PKI) zur Authentifizierung und symmetrischen Verfahren zur Enkryption der Nutzdaten zu unterscheiden, wie sie häufig in den gängigen Protokollen vorkommt und daher nicht als hybrid bezeichnet wird [129]. Außerdem ist der hybride Einsatz von Zertifikaten anzudenken, um Signaturen ebenso zu hybridisieren, wie die Verschlüsselung.

Das BSI empfiehlt in [1] die neuen quantensicheren Verfahren zunächst einmal nur mit klassischen Methoden zusammen einzusetzen. Somit wäre ein fließender Übergang hin zu PQC/ QKD möglich. Von der Kombination verschiedener Schlüssel können sowohl die klassischen Methoden profitieren, als auch die PQC/QKD Methoden bezüglich der Unerprobtheit / Ausfallsicherheit. Eine erhöhte Rechenkomplexität und die damit einhergehende Latenz fällt je nach Komplexität der Verschachtelungen meist gering aus.

Für einen hybriden Schlüsselaustausch bietet sich der Einsatz von QKD an und es gibt mehrere Proof-of-Concepts (POC), die eine Hybridisierung mit QKD vorschlagen, sowie erste Testbed-Realisierungen.

In Abbildung 23 wird der weitere Inhalt des Kapitels gezeigt, der sich sowohl mit hybridem Schlüsselaustausch als auch hybriden Authentifizierungsmethoden beschäftigt.

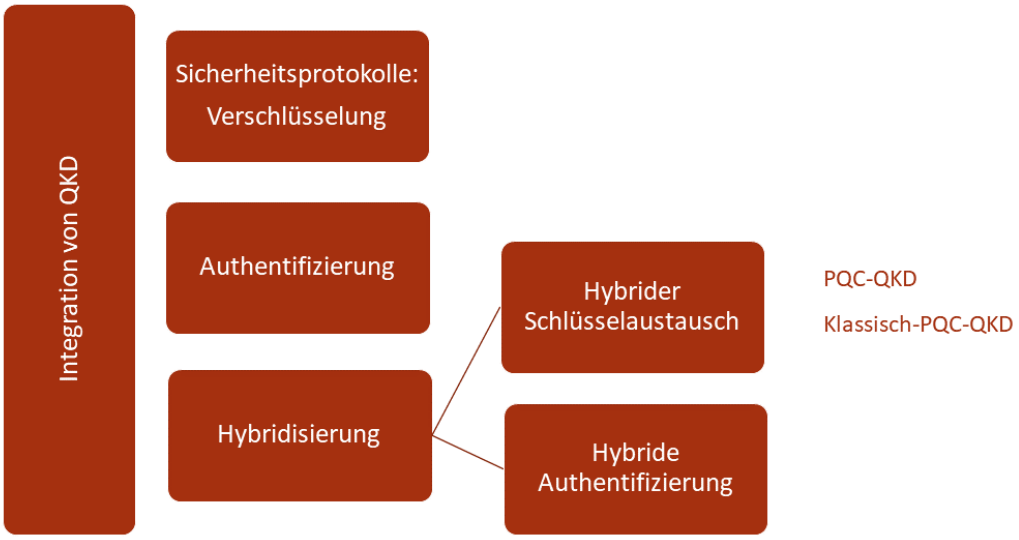


Abbildung 23: Übersicht Kapitel C: Hybridisierung mit QKD im Bereich des Schlüsselaustauschs und der Authentifizierung.

Hybrider Schlüsselaustausch

Es gibt verschiedene Möglichkeiten, wie verschiedene Schlüssel miteinander kombiniert werden können. Die bekannteste ist neben der bloßen Aneinanderreihung die eines einfachen XORs. Darüber hinaus werden häufig KDFs zur Kombination verwendet. Dabei stellen Pseudo-Random Funktionen (PRF) eine ein-schrittige KDF dar. Unter einer zwei-schrittigen KDF versteht man eine Funktion, die erst extrahiert und dann erweitert [130]. Hierzu zählt z.B. die Anwendung des HMACs [131], [132]. Dabei muss beim jeweiligen Protokoll darauf geachtet werden, dass die Hybridisierung auch den Sicherheitsbeweis erfüllt.

Im Weiteren werden verschiedene Arten von hybriden Verschlüsselungen vorgestellt. Dabei werden zunächst Beispiele für die hybride Erweiterung von QKD nur mit PQC dargestellt, bevor sich den Protokollen, die mehrere Verschlüsselungen zusammen mit klassischen Methoden hybrid kombinieren, gewidmet wird.

Die ITU definiert in einem technischen Report die Hybridisierung von Schlüsseln in [133]. Dabei werden v.a. hybride Standards des Layer 3 betrachtet (vgl. Kapitel „Schicht 3"). Hierbei werden mehrere alternative Methoden zum Schlüsselaustausch vorgestellt:

- RFC 8784 (IKEv2 & post-quantum Keys) [134]: jedes IKE-Peer hat feste Liste an PSKs mit IDs und Benachrichtigungen; QKD-Keys können nur maximal bei jeder Erneuerung der IKE SA festgelegt werden und lassen sich somit eher als statische PSK integrieren; hier ist die initiale IKE SA noch nicht durch PPKs geschützt;
- RFC 9242 [135] führt IKE_INTERMEDIATE für die Übertragung großer Datenmengen im initialen Key Exchange ein und vermeidet dadurch unnötige Fragmentierung, während RFC 7383 die Fragmentierung auf IKE-Level statt IP Fragmentierung ermöglicht;
- Der Entwurf *Mixing Preshared Keys in the IKE_INTERMEDIATE and in the CREATE_CHILD_SA Exchanges of IKEv2 for Post-quantum Security* [27] zeigt, wie IKE_INTERMEDIATEs genutzt werden können, um PPKs (QKD-Keys) durch Rekeying in initiale IKE SAs zu integrieren; Ist ein frischer Schlüssel dynamisch während der aktiven Zeit einer IKE SA zur Verfügung, so kann dieser ohne die SA komplett zu erneuern, hinterlegt werden. So wird ein CREATE-CHILD-SA Exchange für das Rekeying der Child SAs gesendet;
- RFC 9370 (Update RFC 7296 IKEv2) [136]: ein oder mehrere alternative Key Exchange Methoden; nur Schlüssel von vereinbarten Key Exchanges werden initial ausgetauscht, weitere werden über Transform Types mit IDs verhandelt; Neben dem Key Exchange des Transform Type 4, der immer in der IKE-SA-INIT erfolgt, müssen zusätzlich verhandelte Key Exchanges in jeweils einzelnen, der Reihenfolge entsprechenden IKE_INTERMEDIATEs stattfinden; dabei spielt es keine Rolle, ob hier ein klassischer Key Exchange oder post-quanten Verfahren ausgehandelt wird, allerdings darf der Key Exchange nur eine Runde betragen;
- ⇨ In RFC 9370 werden für das Rekeying IKE_FOLLOWUP_KEs eingesetzt; somit eignet sich dieser Standard für dynamisch erneuerbare QKD-Keys;

Erste hybride Internet Drafts, die PQC KEM zusammen mit klassischen Verschlüsselungsalgorithmen hybrid einsetzen, existieren auch für TLS ab OpenSSL 3.2 [38] (vgl. Kapitel „Schicht 4 & 5") und SSH [75] (vgl. Kapitel „Secure Shell (SSH)").

Bei ETSI [137] handelt es sich vorwiegend um die Hybridisierung von post-quantum Public Key Exchange Mechanismen, gibt aber vor, dass bei der konkatenierenden KDF und der kaskadierenden KDF die Kombination mit PSK aus QKD kommen kann.

PQC - QKD

Die neuen NIST KEM PQC-Algorithmen (siehe „PQC Digitale Signaturen“) sind einfach implementierbar und werden in Zukunft voraussichtlich verstärkt mit QKD zusammen eingesetzt werden, um die mathematisch relative Sicherheit mit physikalisch feststellbarer Sicherheit zu untermauern. Dieses Unterkapitel befasst sich speziell mit der Kombination von QKD mit PQC-Algorithmen.

Im Gegensatz zu HQC, welcher mit hohen Längen der öffentlichen Schlüssel und Chiffretexten einhergeht, ist ML-KEM (FIFS 203) durch seine geringe Notwendigkeit zur Fragmentierung gut für Hybridisierung geeignet. [138] demonstriert, wie der ML-KEM mit QKD zusammen in IPsec bzw. TLS eingesetzt wird. Dabei wird der parallele gegenüber dem sequentiellen hybriden Einsatz von QKD und PQC diskutiert. Der parallele Ansatz bringt den Vorteil, dass sich die Latenzen nicht aufmultiplizieren. Tatsächlich bringt der parallele Ansatz im Vergleich zu reiner QKD keinen extra Delay, der sequentielle Ansatz dagegen durch seine zwei extra Runden durch IKE_INTERMEDIATE etwa 200 ms. Auch QKD an sich produziert leichten Overhead durch den Key-ID-basierten Abruf, aber nicht so großen, wie PQC, was leicht zum Bedarf an Fragmentierung führt. Hier wird die Schlüsselableitung komplett mit QKD ersetzt. Ähnlich zum abgelaufenen Entwurf, der sich mit der Abänderung von IPsec (IKEv2) zur Integration von QKD beschäftigt [28], wird dieser mit der sequentiellen Hybridisierung von RFC 9370 erweitert, um schließlich mit einer hybriden PQC-Encapsulation den QKD-Key zu umfassen. Ein Protokollablauf initiiert vom Server [vom Client] mittels ETSI 004/014 APIs (Indexmanagement für Rekeying noch ausbaufähig) sieht folgendermaßen aus [138]:

1. Hybride Schlüsselgenerierung (Alice):
   Der Initiator generiert gleichzeitig ein PQC-Schlüsselpaar [und wenn er Client ist, ruft er zusätzlich eine QKD-Key-ID aus der KME via ETSI-API ab]
2. IKE_SA_INIT Request:
   Alice überträgt den öffentlichen PQC-Schlüssel [und auch die QKD-Key_ID] in einer [gemeinsamen] KEY_EXCHANGE Payload
3. Hybride Encapsulation (Bob):
   Der Responder ruft eine QKD-Key-ID [bzw. den entsprechenden QKD-Key anhand der empfangenen ID] ab und umkapselt den Chiffretext [und die ID] mit dem öffentlichen PQC-Schlüssel
4. IKE_SA_INIT Response:
   Bob gibt den PQC-encodierten Chiffretext [und die ID] in der KEY_EXCHANGE Payload zurück
5. Hybride Decapsulation (Alice):
   Der Initiator führt eine PQC-Entkapselung durch und ruft den Quantenschlüssel unter Verwendung der [zuvor] erhaltenen ID ab
6. Schlüsselableitung:
   Beide Parteien verketten die gemeinsamen Geheimnisse: Shared_secret = PQC_secret | QKD_key; Das Geheimnis bleibt solange sicher, bis beide Verschlüsselungen gebrochen wurden, während das Geheimnis der Sicherheit des Algorithmus der stärksten Komponente entspricht!

Der wesentliche Unterschied besteht darin, dass der Client-initiierte Ablauf im Request den öffentlichen PQC-Schlüssel neben der QKD-KEY-ID sendet, während beim Server-initiierten Ablauf im Response erst die QKD-Key-ID zusammen mit dem Chiffretext gesendet wird.

Eine ableitende Integration QKD für ML-KEM Kyber wird in [139] behandelt. Aus dem QKD-Key wird der PQC-Seed für den öffentlichen bzw. privaten Schlüssel abgeleitet. Nur die daraus erhaltenen PQC-Schlüssel werden zur Nachrichtenverschlüsselung eingesetzt. Dabei darf die QKD-Key-Länge 20 QuBit nicht übersteigen, da sich sonst die Laufzeit des hybriden Algorithmus gegenüber dem reinen Kyber verlängern würde.

Eine experimentelle Integration von QKD und PQC wird sowohl zum Schlüsselaustausch als auch zur Authentifizierung in [129] vorgestellt. Dabei wird zur Verschlüsselung QKD mit einer Post-Quanten-KEM (PQ-KEM) und einer klassischen KEM mit einer KDF (HMAC) integriert. Zur Authentifizierung wird ebenfalls ein Post-Quanten-DSA (PQ-DSA) und ein klassischer Informationstheoretisch-sicherer MAC eingesetzt (Weiteres siehe Kapitel „Hybride Authentifizierung"). Zusätzlich wird eine Physical Unclonable Function (PUF), welche einem einzigartigen intrinsischen Fingerabdruck des Gerätes gleichkommen, zum Maskieren des geheimen Zustandes eingesetzt. Nur eine Ausführung auf demselben physischen Gerät kann dieses Geheimnis wieder demaskieren. In die KDF fließen das QKD-Material, sowie die Schlüssel aus der (PQC)-KEM. Dabei wird das QKD-Key-Material als geteiltes Array an informationstheoretisch sicheren zufälligen Bits angesehen. Die PRF wird anschließend für beide Nutzer in einen geheimen Zustand für diese und die nächste Sitzung aufgeteilt. Genau wie beim folgenden Experiment, wird das T12-QKD-Protokoll eingesetzt.

[105] zeigt für Toshiba's QKD-Protokoll, das T12-Protokoll und PQC Dilithium-Digital Signature Algorithmus (DSA) eine erfolgreiche Hybridisierung. T12 ist eine Modifikation des Standard-BB84-Protokolls mit Decoy-States, bei dem die Wahrscheinlichkeit, dass Bitwerte in der Informationsbits-erzeugenden Z-Basis kodiert werden, höher ist. Dadurch wird die SKR unter vielen Bedingungen nahezu verdoppelt. Das Paper zielt darauf ab, verschiedene noch-nicht gängige Ansätze miteinander zu kombinieren. So wird bei der Verschlüsselung neben AES auch ChaCha20 untersucht. Letzterer bringt als Stream-Cipher deutliche Geschwindigkeitsvorteile gegenüber der Block-Cipher AES mit. Details zu in dieser Arbeit ebenfalls eingesetzten hybriden Authentifizierungsprotokollen, finden sich in Kapitel „Hybride Authentifizierung". Insgesamt soll eine adaptive hybride Struktur für effiziente Protokolle geschaffen werden.

Klassisch - PQC - QKD

QKD wird meist auch mit mehreren Verfahren – typischerweise mit PQC und klassischen (sowohl asymmetrische und symmetrische) Algorithmen – zusammen eingesetzt. Im Vergleich sind solche Methoden, die auf der Kombination von mehreren Schlüsseln basieren am besten geschützt vor Angriffen, die einen Mangel an einer Schlüsselressource provozieren, um zu erzwingen, dass die Daten schlechter bzw. unverschlüsselt übertragen werden. Es muss dennoch bestmöglichst verhindert werden, dass Angreifer eine Schlüsselressource zum Leerlaufen bringen, sodass die Sicherheit der übrigen Schlüssel reduziert wird. Beim Design ist zudem darauf zu achten, dass sich die einzelnen Verfahren nicht negativ beeinflussen und so das Sicherheitsniveau auf das schwächste Verfahren abfällt, was in der Regel die klassische Verschlüsselung darstellt. Zudem müssen die neuen Protokolle sowohl mit eventuellen Vorgängerversionen abwärtskompatibel sein, aber dennoch der forcierten Downgrade-Attacke widerstehen können.

Es gibt auch spezielle Hardware (z.B. vom BSI zugelassene der Firma Adva, siehe Kapitel „Schicht 1“) bei der bereits eine Hybridisierung mit QKD-Keys mittels einer KDF implementiert ist. Das Besondere an diesen Geräten ist, dass sie selbst über klassische (DH) / PQC-Verschlüsselungsmethoden verfügen und so bei Schlüsselnotstand auch ganz auf diese zurückgegriffen werden kann. [9] verwendet diese, um eine KDF aus klassischen, QKD, und PQC-Schlüsseln zu realisieren (siehe Abbildung 24). Im Adva-Transponder wird der eingespeiste globale Schlüssel mit einem intern generierten DH-Key (DHK) zu einem gemischten Schlüssel berechnet und der eingehende Datenstrom damit verschlüsselt.

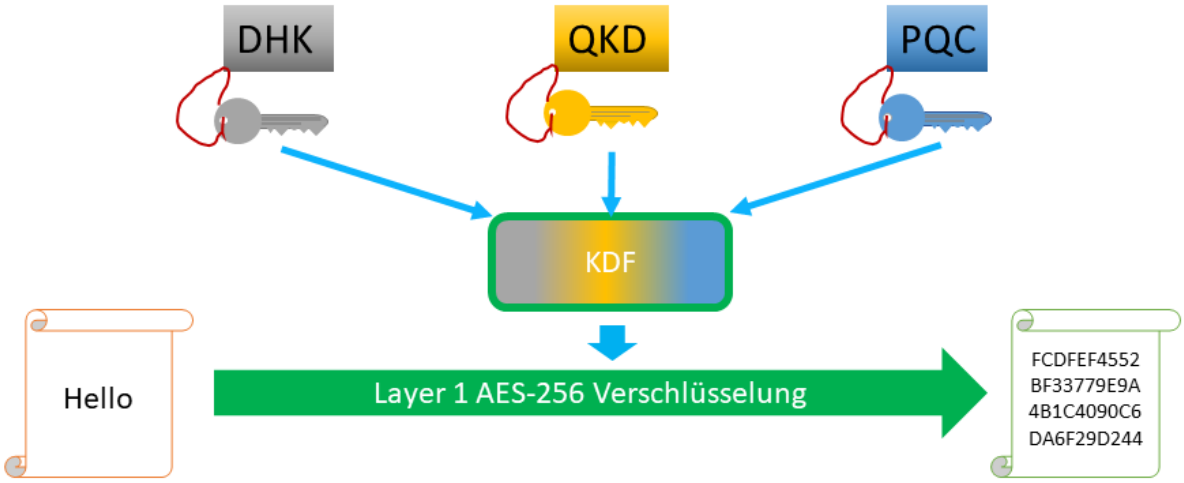


Abbildung 24: Key Derivation Function (KDF) mit klassischem, QKD- & PQC-Schlüssel: Kombination aus drei verschiedenen Schlüsseln von unterschiedlichen Quellen (DH-, QKD- und PQC- Keys) zur Layer 1 Verschlüsselung [9].

In [140] wird ein typischer PRF-then-XOR split-key Combiner für DH, PQC & QKD vorgestellt. Für die PRF wird meistens HMAC verwendet, während sich v aus DH-privaten Schlüsseln von Alice und Bob, dem Chiffretext und dem Universally Unique Identifier (UUID) zusammensetzt, wie Gleichung (2) zeigt.

$$K \leftarrow \mathrm{PRF}(k_{dh}, v) \oplus \mathrm{PRF}(k_{pqc}, v) \oplus \mathrm{PRF}(k_{qkd}, v) \quad \text{mit } v \leftarrow dhpk_A \parallel dhpk_B \parallel c \parallel uuid \tag{2}$$

[141] zeigt die Kombination von QKD und PQC mit klassischen Keys. Die Schlüsselkombination nimmt nicht viel Rechenleistung in Anspruch und wird auf einem FPGA ausgeführt. Es wir als klassischer Algorithmus ECDH eingesetzt, eine authentifizierte Version von CRYSTALS-KYBER als PQC-KEM mit Langzeit-Pre-Shared Public Keys und ein QKD-Gerät von IDQ. Als Konstruktionsvorlage für die KDF dient SHA-3-512 und AES-256-GCM. Die Schlüsselrate kann bis zu 1624 keys/s betragen, wenn kein QKD eingesetzt wird. Bei Verwendung von QKD allerdings sinkt die Rate rund auf das 180-stel. QKD ist somit in dieser Implementierung wesentlich langsamer als ECDH und KYBER.

Beim Berliner Testbed OpenQKD [142] wird beispielsweise für eine zukunftssichere Ende-zu-Ende-Verschlüsselung ein pseudo-zufälliger Nutzerschlüssel hybrid mit PQC verschlüsselt, um ihn bei der QKD-Verschlüsselung auch bei der Entschlüsselung an Trusted Nodes zu schützen (siehe Abbildung 25).

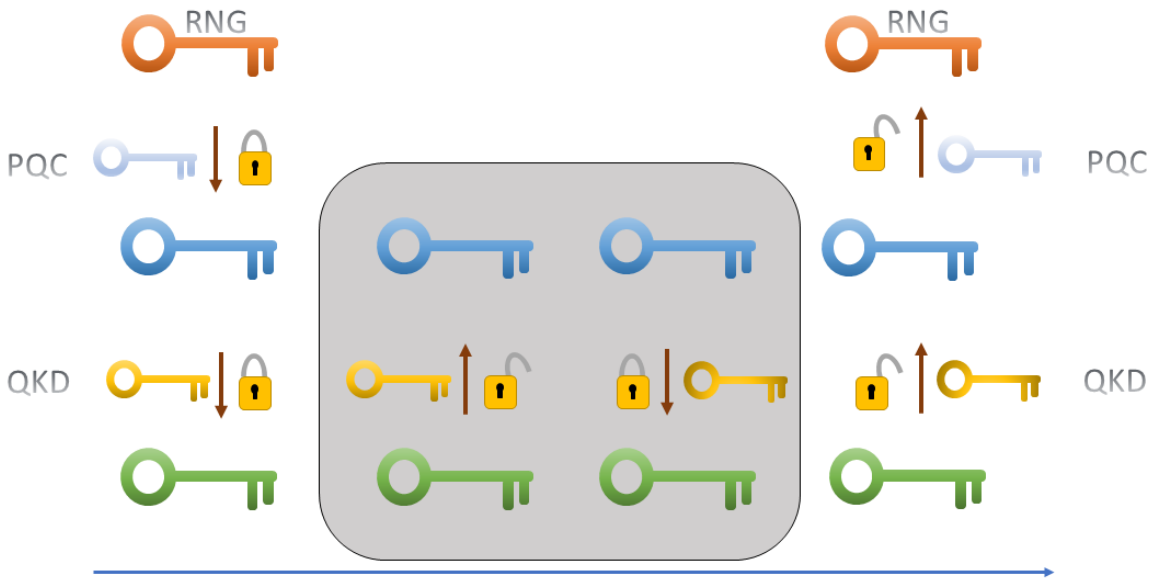


Abbildung 25: Hybrider Key Exchange am Beispiel des Berliner Testbeds OpenQKD [142]: Ein zufällig generierter Schlüssel (RNG, orange) wird mittels PQC-Key (silber) Ende-zu-Ende verschlüsselt (blau), damit er bei der abschnittsweise unterbrochenen QKD-Verschlüsselung (grün) mit QKD-Key (gold) auch am Trusted Node (grauer Kasten) verschlüsselt vorliegt (blau).

Die vom BSI zugelassene, kommerzielle Layer 1 Verschlüsselungssystem-FSP-3000-Serie von Adva [12] (siehe Kapitel „Schicht 1“), ermöglicht eine hybride Kombination von klassischen und PQC-Schlüsseln. Auch eine Einspeisung von QKD-Keys ist möglich.

Für hybride Verfahren eignet sich ebenfalls die Kombination von Secret Sharing mit verschiedenen Key Exchange Protokollen (QKD-, PQC-basiert), die auf unterschiedlichen Links mit den Teilgeheimnissen via XOR verknüpft werden [143]. Statt auf klassisches Linear-Code-basiertes Secret Sharing zu setzen, könnte dieses auch durch quantenbasiertes QSS ersetzt werden, welches in Kapitel „Quantum Secret Sharing (QSS), Threshold-based (TQSS) & Quantum Conference Key Agreement (QCKA“ vorgestellt wurde.

Das Muckle-Protokoll [144] kombiniert in einem Hybriden Authentifizierten Key Exchange (HAKE) Schlüssel aus einem QKD-Protokoll (simuliert) mit dem von einem PQC-KEM (Supersingular Isogeny-based Diffie-Hellman) und einem klassisch sicheren KEM (ECDH). Das Protokoll wurde als Grundlage für Erweiterungen genommen. Das Muckle+ Protokoll [145], was zur Authentifizierung eine digitale Signatur und eine PKI einsetzt, unterscheidet sich vom Muckle# [146], welches statt der Signatur einen PQC-KEM zur Authentifizierung einsetzt. Bei Muckle# wird zur Verschlüsselung eine Kombination aus klassischem KEM, PQC-KEM, MAC, PRF und ein QKD-Key genutzt. Das Muckle Protokoll [147] bietet auch Ende-zu-Ende-Verbindungen zur Absicherung von ganzen Netzwerken mit PSK oder PQC-Signaturen.

Die Gruppe ETSI STF 684 gefördert von der EU-Kommission strebt bis Ende 2027 eine Standardisierung mit Sicherheitsbeweis von hybriden Verfahren mit PQC und QKD an [148].

Hybride Authentifizierung

In diesem Abschnitt werden speziell Authentifizierungsverfahren behandelt, die auf hybrider Schlüsselkombination basieren. Der Vorteil der Kombination von quanten-basierten mit klassischen Verfahren liegt hier ebenfalls im flexiblen Einsatz zur optimalen Nutzung der Vorteile beider Technologien. So kann z.B. der Bedarf an Quantenspeichern durch den Einsatz von zusätzlicher klassischer Kommunikation umgangen werden und neue quantenbasierte Ansätze auf ihre Praxistauglichkeit geprüft werden, während man immer noch auf die bewährten klassischen Verfahren zurückgreifen kann. Es folgen einige Beispiele aus der Literatur.

Das oben vorgestellte Muckle-Protokoll [144] setzt zur Authentifizierung QKD-Keys als PSKs mittels HMAC ein. Der berechnete MAC-tag basiert zusätzlich jedes Mal auf einem neu abgeleiteten Schlüssel. Somit kann der mit PQC-Signaturen verbundene Overhead umgangen werden. Zusätzlich wird der normale PSK für die Authentifizierung der QKD-Partner benötigt.

Eine experimentelle Integration von QKD und PQC wird sowohl zur Authentifizierung als auch zum Schlüsselaustausch in [129] vorgestellt. PQC kann die Skalierungsprobleme, die manuelle Authentifizierung mit PSKs bei QKD mit sich bringt, kompensieren, indem zusätzlich klassische bzw. PQC-Digitale Signaturen eingesetzt werden. Sind keine QKD-PSKs zur Authentifizierung vorhanden, so werden PQC- und klassische Digitale Signaturen eingesetzt. In diesem Fall ist das Protokoll nicht mehr informationstheoretisch-sicher, dafür aber noch quanten-sicher. Ist ein QKD-PSK vorhanden, so wird die Nachricht mit dem informationstechnisch-sicheren MAC authentifiziert.

Es kann auch ein PQC-Verfahren (Falcon) zum Absichern der Nachverarbeitung in den klassischen Kanälen der QKD-Schlüsselgenerierung selbst in QKD-Geräten eingesetzt werden, wie [149] demonstriert. Die Authentifizierung wird bereits bei der Sifting Phase der Schlüssel durchgeführt, um die anschließende Reconciliation der Schlüssel zu verschlüsseln. So könnte mehr QKD-Schlüsselmaterial erhalten werden, da beim anschließenden Privacy Amplification-Prozess nicht so viele Bits offen gelegt sein könnten. Allerdings ist ein Experimentieren mit kommerzieller Hardware (statt wie im Paper nur anhand von Simulationen), nur in enger Zusammenarbeit mit den jeweiligen Herstellern oder an eigens entwickelten QKD-Geräten möglich. Ein weiteres Beispiel dafür ist [106]. Hier werden PSKs oder ein PQC-Algorithmus zur Authentifizierung der einzelnen QKD-Nachverarbeitungsschritte angewandt. Die erste Authentifizierung nach dem Basis Sifting wird einmal pro Sekunde durchgeführt; gefolgt von der Authentifizierung nach der Error Correction, dem Zufallszahlenaustausch und der Privacy Amplification. Der Vorteil von PQC ist, dass geringe Rechenressourcen benötigt werden und so eine Authentifizierung nur 1 ms in Anspruch nimmt.

In [150] wird für digitale Signaturen und deren Verifizierungen CRYSTALS-Dilithium3 eingesetzt, während für den authentifizierten Schlüsselaustausch klassisches ECDH-Ephemeral (ECDHE) in Kombination mit CRYSTALS-Kyber eingesetzt wird. Zusätzlich wird QKD-Schlüsselmaterial zur Absicherung verwendet. In OpenSSH implementiert beträgt die Dauer eines Handshakes mit allen drei Verschlüsselungsverfahren 62 ms, wobei QKD etwa 12ms ausmacht.

Das OpenQKD Testbed in Berlin zeigt in [142] die Kombination von PSK-Authentifizierung, QKD und PQC mit digitalen Signaturen. Eine kombinierte Authentifizierung auf Basis von ausgetauschten QKD-Keys und PSKs wurde zwar versucht, aber wegen der besseren Skalierung in großen Netzwerken durch die Integration von PQC-Authentifizierung über Zertifikate ersetzt. Das Testbed basiert somit auf zwei PKI mit root CA. Es werden für die klassische PKI RSA und Falcon 1024 für die PQC-basierte PKI eingesetzt. Dies demonstriert gut die praktische Notwendigkeit und den damit verbundenen Mehraufwand von hybriden Verfahren, da nicht alle Geräte dieselben Verfahren unterstützen.

Fazit zur Hybridisierung

Hybride Protokolle werden in der Übergangszeit zu quantensicheren Algorithmen eine große Rolle spielen, da sie gerade beim Einsatz von mehreren quantensicheren Protokollen in Kombination zusätzliche Redundanz bieten und so eine flexible Anpassung an die jeweils benötigten Anforderungen bieten. Hybridisierung ist ein weiter Begriff und kann in der Praxis meist unterschiedliche Bedeutungen annehmen: Von einer hybriden Schlüsselnutzung bis zu hybriden Protokollen oder die hybride Authentifizierung mit hybriden Signaturen.

Den hybriden Schlüsselaustausch ermöglichen erste Standards für Layer 3/5. Dazu ist eine KDF nötig, die die verschiedenen Schlüssel miteinander kombiniert. Es ist auch eine Verkapselung der Schlüssel möglich, bei der ein Schlüssel mit einem anderen Schlüssel, der aus einem anderen Verfahren stammt, verschlüsselt wird. Klassische Verfahren werden meist nur in Kombination mit PQC und QKD eingesetzt. Bei der Implementierung von hybriden Protokollen ist jeweils die Sicherheitsprüfung und die Dauer der hybriden Ver- & Entschlüsselung zu berücksichtigen. Bei der hybriden Authentifizierung mit QKD wird meist eine Kombination aus einem QKD-PSK für einen MAC mit PQC/ klassischen Signaturen ermöglicht. Steht kein QKD-Key zur Verfügung, kann das System auf digitale Signaturen zurückgreifen. Der Vorteil von QKD-PSK-basierten Verfahren ist, dass keine zusätzliche PKI nötig ist.

Zudem bietet die Redundanz hybrider Methoden auch Ausfallsicherheit bei DDoS Attacken, welche es auf das Leerlaufen des Schlüsselmaterials abgesehen haben. Bisher gibt es eher wenige hybride Schlüsselaustauschverfahren, die auf QKD und PQC setzen, dagegen scheinen Kombinationen mit klassischen Verfahren von größerem Interesse. Je nach Anwendungsfall kann die Art der Schlüsselkombinationen sehr unterschiedlich ausfallen.

Fazit zur sicheren QKD-Schlüsselintegration

Das Dokument zeigt verschiedene Einsatzgebiete von QKD-Keys. Der Einsatz in verschiedene Sicherheitsprotokolle wird auf allen Layern diskutiert. Es wird demonstriert, dass eine Verschlüsselung mit QKD auf jedem Layer erfolgen kann, auch wenn es sinnvoll ist, möglichst auf niedrigen Layern (meist 2/3) zu verschlüsseln, da die höheren somit automatisch geschützt sind. Dabei ergeben sich abhängig vom Protokoll verschiedene Einsatzmöglichkeiten. Meist lassen sich QKD-Keys als Pre-shared Keys verwenden. Gerade zu Authentifizierungszwecken lassen sich diese einfach im Vorfeld verteilen. Schwieriger ist es, die QKD-Keys in die Verschlüsselungsalgorithmen selbst zu integrieren. Meist werden verschiedene Schlüssel verwendet, wie Langzeit- oder davon abgeleitete Kurzzeitschlüssel. Eine Wahl der richtigen Schlüssellebenszeiten, welche sich wiederum nach der zur Verfügung stehenden Schlüsselrate richtet, ist ebenso von Bedeutung, wie ein effizientes, zeitkritisches Routing und Key Management im Netzwerk, um die Schlüssel auch zeitnah beim Kommunikationspartner zur Verfügung zu haben.

Die quantensichere Authentifizierung, die natürlich einen festen Teil der Sicherheitsprotokolle bildet. Es wird verdeutlicht, dass auch klassische Protokolle wie Kerberos oder MAC mit QKD-Keys verwendet werden können, auch wenn die Authentifizierung nicht langzeit-sicher über die Beendigung der Sitzung hinaus sein muss. Zum jetzigen Zeitpunkt eignen sich Digitale Signaturen (CRYSTALS-Dilithium, SPHINCS+ und Falcon) mit ausreichender Rechenkomplexität. Rein quantenbasierte Verfahren sind noch theoretisch und bekommen erst durch kommerziell verfügbare Quantenspeicher in Zukunft Bedeutung.

Um die Integration von QKD langsam in der Praxis zu realisieren, bieten sich hybride Ansätze zur Verschlüsselung mit KDFs aus (klassischen), PQC und QKD-Keys an. Das Konzept der Hybridisierung lässt sich auch auf die Authentifizierung anwenden. Bei der hybriden Authentifizierung mit QKD trifft man meist auf eine Verbindung mit PQC-Digitalen Signaturen und Pre-Shared QKD Keys mit MACs.

Eine Demonstration dieser verschiedenen aufgezeigten, teilweise parallel eingesetzten Möglichkeiten zur Integration von QKD-Keys in unterschiedlichen Testbeds, ist für eine Weiterentwicklung der QKD-Integration unerlässlich.

Abbildungsverzeichnis

Literaturverzeichnis

[1] BSI, Ed., “Quantum-safe cryptography – fundamentals, current developments and recommendations,” 2021, [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html?nn=916626

[2] D. I. Bloch and D. A. Kretschmann, “AGENDA QUANTENSYSTEME 2030”, [Online]. Available: https://www.quantentechnologien.de/fileadmin/public/Redaktion/Dokumente/PDF/Publikationen/Agenda_Quantensysteme_2030_web_C1.pdf

[3] “Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,” ITU. Accessed: Sep. 24, 2025. [Online]. Available: https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=2820

[4] *ETSI GS QKD 004 Quantum Key Distribution (QKD); Application Interface*, Aug. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/004/02.01.01_60/gs_QKD004v020101p.pdf

[5] *ETSI GS QKD 014 Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API*, Feb. 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_QKD014v010101p.pdf

[6] “[110] OTN Sec : Security for OTN beyond 100 Gbit/s.” Accessed: Jun. 23, 2025. [Online]. Available: https://www.itu.int/md/T13-SG15-C-0110/en

[7] P. Kampanakis, M. Campagna, E. Crocket, A. Petcher, and S. Gueron, “Practical Challenges with AES-GCM and the need for a new cipher”.

[8] N. Makris *et al.*, “Field demonstration of a fully managed, L1 encrypted 3-node network with hybrid relayed-QKD and centralized symmetric classical key management,” Mar. 13, 2024, *arXiv*: arXiv:2403.08526. doi: 10.48550/arXiv.2403.08526.

[9] E. Pincemin *et al.*, “400-Gbps Coherent Transmission of 100-Gbps QKD-Secured Data Stream Over 184-km of Standard Single Mode Fiber Through Three QKD Links and Two Trusted Nodes,” *Journal of Lightwave Technology*, vol. 42, no. 12, pp. 4302–4309, Jun. 2024, doi: 10.1109/JLT.2024.3397042.

[10] P. Gavignet *et al.*, “Co-Propagation of QKD & 6 Tb/s (60 × 100G) DWDM Channels With ∼17 dBm Total WDM Power in Single and Multi-Span Configurations,” *Journal of Lightwave Technology*, vol. 42, no. 4, pp. 1321–1327, Feb. 2024, doi: 10.1109/JLT.2023.3324840.

[11] “1830 Photonic Service Interconnect – Modular (PSI-M) | Nokia.com.” Accessed: Sep. 22, 2025. [Online]. Available: https://www.nokia.com/optical-networks/1830-psi-m/

[12] “ConnectGuard™ security for optical networks,” advasecurity. Accessed: Jun. 27, 2025. [Online]. Available: https://www.advasecurity.com/en/resources/resources-gated-page/solution-briefs/connectguard-security-for-optical-networks

[13] “IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security,” *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pp. 1–239, Dec. 2018, doi: 10.1109/IEEESTD.2018.8585421.

[14] “IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control,” *IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018)*, pp. 1–289, Feb. 2020, doi: 10.1109/IEEESTD.2020.9018454.

[15] J. Y. Cho and A. Sergeev, “Using QKD in MACsec for secure Ethernet networks,” *IET Quantum Communication*, vol. 2, no. 3, pp. 66–73, 2021, doi: 10.1049/qtc2.12006.

[16] “Konfigurieren von MACsec | Junos OS | Juniper Networks.” Accessed: Sep. 29, 2025. [Online]. Available: https://www.juniper.net/documentation/de/de/software/junos/security-services/topics/task/macsec.html
[17] “Validation of a Quantum Safe MACsec Implementation,” 2022, Accessed: Jun. 04, 2024. [Online]. Available: https://www.juniper.net/content/dam/www/assets/white-papers/us/en/2022/validation-of-quantum-safe-macsec-white-paper.pdf
[18] R. Singh, C. Hill, S. Kawaguchi, and J. Lupo, “Secure Key Integration Protocol (SKIP),” Internet Engineering Task Force, Internet Draft draft-cisco-skip-01, Mar. 2025. Accessed: Mar. 17, 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-cisco-skip-01
[19] “ANYsec.” Accessed: Jun. 30, 2025. [Online]. Available: https://documentation.nokia.com/sr/23-10-2/books/segment-routing-pce-user/anysec.html
[20] E. Dervisevic and M. Mehic, “Overview of Quantum Key Distribution Technique within IPsec Architecture,” Dec. 24, 2021, *arXiv*: arXiv:2112.13105. doi: 10.48550/arXiv.2112.13105.
[21] A. Mink, S. Frankel, and R. Perlner, “Quantum Key Distribution (QKD) and Commodity Security Protocols: Introduction and Integration,” vol. 1, no. 2, 2009, [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901508
[22] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, “Internet Key Exchange Protocol Version 2 (IKEv2),” Internet Engineering Task Force, Request for Comments RFC 7296, Oct. 2014. doi: 10.17487/RFC7296.
[23] “BSI-TR-02102-3.pdf.” Accessed: May 23, 2025. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-3.pdf?__blob=publicationFile&v=8
[24] M. Mehic *et al.*, “Quantum Cryptography in 5G Networks: A Comprehensive Overview,” *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 302–346, 2024, doi: 10.1109/COMST.2023.3309051.
[25] S. Marksteiner and O. Maurhart, “A Protocol for Synchronizing Quantum-Derived Keys in IPsec and its Implementation,” 2015.
[26] M. A. Sfaxi, S. Ghernaouti-Hélie, G. Ribordy, and O. Gay, “Using Quantum Key Distribution within IPSEC to secure MAN communications,” Jan. 2005, [Online]. Available: https://www.researchgate.net/publication/236200262
[27] V. Smyslov, “Mixing Preshared Keys in the IKE_INTERMEDIATE and in the CREATE_CHILD_SA Exchanges of IKEv2 for Post-quantum Security,” Internet Engineering Task Force, Internet Draft draft-ietf-ipsecme-ikev2-qr-alt-10, May 2025. Accessed: Aug. 07, 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ipsecme-ikev2-qr-alt
[28] S. Nagayama and R. V. Meter, “IKE for IPsec with QKD,” Internet Engineering Task Force, Internet Draft draft-nagayama-ipsecme-ipsec-with-qkd-01, Oct. 2014. Accessed: Aug. 07, 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-nagayama-ipsecme-ipsec-with-qkd-01
[29] “QKD Schlüsselmanagement,” muquanet. Accessed: Jun. 28, 2024. [Online]. Available: https://www.unibw.de/muquanet/projekte/qkd-managementsystem
[30] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” Internet Engineering Task Force, Request for Comments RFC 8446, Aug. 2018. doi: 10.17487/RFC8446.
[31] B. A. Hubermann, B. Lund, and J. Wang, “Quantum Secured Internet Transport,” *Inf Syst Front*, vol. 22, no. 6, pp. 1561–1567, Dec. 2020, doi: 10.1007/s10796-020-10086-5.
[32] B. Rijsman, “Quantum Key Distribution (QKD) in OpenSSL,” Quantum Key Distribution (QKD) in OpenSSL. Accessed: Sep. 17, 2025. [Online]. Available: https://brunorijsman.github.io/openssl-qkd/doc/implementing-qkd-in-openssl.html
[33] C. Rubio García *et al.*, “Quantum-resistant Transport Layer Security,” *Computer Communications*, vol. 213, pp. 345–358, Jan. 2024, doi: 10.1016/j.comcom.2023.11.010.
[34] T. Prevost, B. Martin, and O. Alibart, “An ETSI GS QKD compliant TLS implementation⋆”.
[35] P. Yan and N. Yu, “The QQUIC Transport Protocol: Quantum-Assisted UDP Internet Connections,” *Entropy (Basel)*, vol. 24, no. 10, p. 1488, Oct. 2022, doi: 10.3390/e24101488.

[36] C. R. Garcia, A. Cano, J. J. V. Olmos, S. Rommel, and I. T. Monroy, “Integrating Quantum Key Distribution into TLS 1.3: A Transport Layer Approach to Quantum-Resistant Communications in Optical Networks,” in *2024 Optical Fiber Communications Conference and Exhibition (OFC)*, Mar. 2024, pp. 1–3. Accessed: Aug. 09, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10527171/?arnumber=10527171&tag=1

[37] J. Blanco-Romero, P. O. García, D. Sobral-Blanco, F. A. Mendoza, A. F. Vilas, and R. P. Díaz-Redondo, “QKD-KEM: Hybrid QKD Integration into TLS with OpenSSL Providers,” Mar. 10, 2025, *arXiv*: arXiv:2503.07196. doi: 10.48550/arXiv.2503.07196.

[38] D. Stebila, S. Fluhrer, and S. Gueron, “Hybrid key exchange in TLS 1.3,” Internet Engineering Task Force, Internet Draft draft-ietf-tls-hybrid-design-14, Jul. 2025. Accessed: Jul. 31, 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design

[39] A.-B. Popa and P. G. Popescu, “Optimal key forwarding strategy in QKD behaviours,” *Scientific Reports*, vol. 14, no. 1, p. 13977, 2024, doi: 10.1038/s41598-024-64994-6.

[40] Md. Armanuzzaman, K. Md. R. Alam, Md. M. Hassan, and Y. Morimoto, “A secure and efficient data transmission technique using quantum key distribution,” in *Proceedings of 2017 4th International Conference on Networking, Systems and Security (NSysS)*, Piscataway, NJ: IEEE, 2017, pp. 1–5. doi: 10.1109/NSYSS2.2017.8267797.

[41] M. Niemiec and P. Machnik, “Authentication in virtual private networks based on quantum key distribution methods,” *Multimed Tools Appl*, vol. 75, no. 17, pp. 10691–10707, Sep. 2016, doi: 10.1007/s11042-014-2299-1.

[42] O. Alia, A. Huang, H. Luo, O. Amer, M. Pistoia, and C. Lim, “100 Gbps Quantum-safe IPsec VPN Tunnels over 46 km Deployed Fiber.” May 07, 2024. [Online]. Available: http://arxiv.org/pdf/2405.04415

[43] J. S. Buruaga *et al.*, “VPN Protection with QKD-Derived Keys Using Standard Interfaces,” in *2023 23rd International Conference on Transparent Optical Networks (ICTON)*, Bucharest, Romania: IEEE, Jul. 2023, pp. 1–4. doi: 10.1109/ICTON59386.2023.10207212.

[44] A. Ghilen, M. Azizi, and R. Bouallegue, “Q-OpenVPN: A New Extension of OpenVPN Based on a Quantum Scheme for Authentication and Key Distribution,” in *Cryptology and Network Security*, vol. 9476, M. Reiter and D. Naccache, Eds., in Lecture Notes in Computer Science, vol. 9476. , Cham: Springer International Publishing, 2015, pp. 238–247. doi: 10.1007/978-3-319-26823-1_17.

[45] J. A. Donenfeld, “WireGuard: Next Generation Kernel Network Tunnel,” in *Proceedings 2017 Network and Distributed System Security Symposium*, San Diego, CA: Internet Society, 2017. doi: 10.14722/ndss.2017.23160.

[46] Y. Nir and A. Langley, “ChaCha20 and Poly1305 for IETF Protocols,” RFC Editor, RFC7539, May 2015. doi: 10.17487/RFC7539.

[47] D. McGrew, K. Igoe, and M. Salter, “Fundamental Elliptic Curve Cryptography Algorithms,” RFC Editor, RFC6090, Feb. 2011. doi: 10.17487/rfc6090.

[48] L. Chen, K. Xue, J. Li, Z. Li, and N. Yu, “Security-Enhanced WireGuard Protocol Design Using Quantum Key Distribution,” in *2024 International Conference on Computing, Networking and Communications (ICNC)*, Big Island, HI, USA: IEEE, Feb. 2024, pp. 718–723. doi: 10.1109/ICNC59896.2024.10556292.

[49] SURF, “Home - Documentation.” Accessed: Aug. 14, 2025. [Online]. Available: https://docs.eduvpn.org/server/v3/

[50] GÉANT, “Home - eduroam.org.” Accessed: Aug. 14, 2025. [Online]. Available: https://eduroam.org/

[51] Klaas Wierenga and Licia Florio, “Eduroam: past, present and future”, doi: 10.12921/CMST.2005.11.02.169-173.

[52] A. Halbouni, L.-Y. Ong, and M.-C. Leow, “Wireless Security Protocols WPA3: A Systematic Literature Review,” *IEEE Access*, vol. 11, pp. 112438–112450, 2023, doi: 10.1109/ACCESS.2023.3322931.

[53] J. C. Klensin, “Simple Mail Transfer Protocol,” Internet Engineering Task Force, Request for Comments RFC 5321, Oct. 2008. doi: 10.17487/RFC5321.
[54] P. E. Hoffman, “SMTP Service Extension for Secure SMTP over Transport Layer Security,” Internet Engineering Task Force, Request for Comments RFC 3207, Feb. 2002. doi: 10.17487/RFC3207.
[55] J. Fenton, “SMTP Require TLS Option,” Internet Engineering Task Force, Request for Comments RFC 8689, Nov. 2019. doi: 10.17487/RFC8689.
[56] “E-Mail-Verschlüsselung in der Praxis,” Bundesamt für Sicherheit in der Informationstechnik. Accessed: Mar. 25, 2026. [Online]. Available: https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Onlinekommunikation/Verschluesselt-kommunizieren/E-Mail-Verschluesselung/E-Mail-Verschluesselung-in-der-Praxis/e-mail-verschluesselung-in-der-praxis.html?nn=131764
[57] K. Moore and C. Newman, “Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access,” Internet Engineering Task Force, Request for Comments RFC 8314, Jan. 2018. doi: 10.17487/RFC8314.
[58] C. Newman, “ESMTP and LMTP Transmission Types Registration,” Internet Engineering Task Force, Request for Comments RFC 3848, Jul. 2004. doi: 10.17487/RFC3848.
[59] H. Tschofenig and P. Eronen, “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS),” Internet Engineering Task Force, Request for Comments RFC 4279, Dec. 2005. doi: 10.17487/RFC4279.
[60] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, “TLS Encrypted Client Hello,” Internet Engineering Task Force, Request for Comments RFC 9849, Mar. 2026. doi: 10.17487/RFC9849.
[61] H. Finney, L. Donnerhacke, J. Callas, R. L. Thayer, and D. Shaw, “OpenPGP Message Format,” Internet Engineering Task Force, Request for Comments RFC 4880, Nov. 2007. doi: 10.17487/RFC4880.
[62] P. Wouters, D. Huigens, J. Winter, and N. Yutaka, “OpenPGP,” Internet Engineering Task Force, Request for Comments RFC 9580, Jul. 2024. doi: 10.17487/RFC9580.
[63] “Technischer Hintergrund › GnuPG › Wiki › ubuntuusers.de.” Accessed: Mar. 12, 2026. [Online]. Available: https://wiki.ubuntuusers.de/GnuPG/Technischer_Hintergrund/
[64] B. C. Ramsdell, “Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification,” Internet Engineering Task Force, Request for Comments RFC 3851, Jul. 2004. doi: 10.17487/RFC3851.
[65] C. Döberl, W. Eibner, S. Gärtner, M. Kos, F. Kutschera, and S. Ramacher, “Quantum-resistant End-to-End Secure Messaging and Email Communication: SMTP,” in *ARES 23*, in ACM Digital Library. New York, NY: The Association for Computing Machinery, 2023, pp. 1–8. doi: 10.1145/3600160.3605049.
[66] fernet, “Fernet Spec,” GitHub. Accessed: Mar. 26, 2026. [Online]. Available: https://github.com/fernet/spec/blob/master/Spec.md
[67] R. Siemborski and A. Melnikov, “SMTP Service Extension for Authentication,” Internet Engineering Task Force, Request for Comments RFC 4954, Jul. 2007. doi: 10.17487/RFC4954.
[68] J. C. Klensin and R. Gellens, “Message Submission for Mail,” Internet Engineering Task Force, Request for Comments RFC 6409, Nov. 2011. doi: 10.17487/RFC6409.
[69] M. Barhoumi, “Quantum-assisted network time synchronisation: A Literature Review, considering examples and challenges,” 2025, *SSRN*. doi: 10.2139/ssrn.5170022.
[70] M. Lipiński, T. Włostowski, J. Serrano, and P. Alvarez, “White rabbit: a PTP application for robust sub-nanosecond synchronization,” in *Control and Communication 2011 IEEE International Symposium on Precision Clock Synchronization for Measurement*, Sep. 2011, pp. 25–30. doi: 10.1109/ISPCS.2011.6070148.
[71] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pp. 1–499, Jun. 2020, doi: 10.1109/IEEESTD.2020.9120376.

[72] A. Meda *et al.*, “QKD protected fiber-based infrastructure for time dissemination,” *Sci Rep*, vol. 15, no. 1, p. 13419, Apr. 2025, doi: 10.1038/s41598-025-97480-8.
[73] S. Asadi and H. S. Shahhoseini, “Formal security analysis of authentication in SNMPv3 protocol by an automated tool,” in *6th International Symposium on Telecommunications (IST)*, Tehran, Iran: IEEE, Nov. 2012, pp. 1060–1064. doi: 10.1109/ISTEL.2012.6483143.
[74] “BSI-TR-02102-4.pdf.” Accessed: Jul. 14, 2025. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-4.pdf?__blob=publicationFile&v=8
[75] P. Kampanakis, D. Stebila, and T. Hansen, “PQ/T Hybrid Key Exchange in SSH,” Internet Engineering Task Force, Internet Draft draft-ietf-sshm-mlkem-hybrid-kex-02, Apr. 2025. Accessed: Jul. 14, 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sshm-mlkem-hybrid-kex
[76] H. Touil, F. Hdioud, N. El Akkad, and K. Satori, “The Security of SSH Protocol Public Key Sharing in the Post-Quantum Era,” *IJC*, pp. 317–323, Oct. 2024, doi: 10.47839/ijc.23.3.3650.
[77] “Quantum Safe Cryptography and Security - An introduction, benefits, enablers and challenges,” ETSI, White Paper, Jun. 2015. Accessed: Jul. 17, 2025. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf
[78] C. M. Lonvick and T. Ylonen, “The Secure Shell (SSH) Transport Layer Protocol,” Internet Engineering Task Force, Request for Comments RFC 4253, Jan. 2006. doi: 10.17487/RFC4253.
[79] M. Pattaranantakul, K. Sanguannam, P. Sangwongngam, and C. Vorakulpipat, “Efficient Key Management Protocol for Secure RTMP Video Streaming toward Trusted Quantum Network,” *ETRI Journal*, vol. 37, no. 4, pp. 696–706, 2015, doi: 10.4218/etrij.15.0114.0883.
[80] D. Zhu, J. Zheng, H. Zhou, J. Wu, N. Li, and L. Song, “A Hybrid Encryption Scheme for Quantum Secure Video Conferencing Combined with Blockchain,” *Mathematics*, vol. 10, no. 17, p. 3037, Aug. 2022, doi: 10.3390/math10173037.
[81] “RFC 6189 - ZRTP: Media Path Key Agreement for Unicast Secure RTP.” Accessed: Sep. 09, 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6189
[82] “RFC 3711 - The Secure Real-time Transport Protocol (SRTP).” Accessed: Sep. 09, 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3711
[83] “ZRTP 4 J Message Codes | Jitsi.” Accessed: Sep. 10, 2025. [Online]. Available: https://desktop.jitsi.org/Documentation/ZRTP4JMessageCodes.html
[84] L. Ferreira and J. Pascal, “Post-quantum Secure ZRTP,” in *Post-Quantum Cryptography*, M.-J. Saarinen and D. Smith-Tone, Eds., Cham: Springer Nature Switzerland, 2024, pp. 3–36. doi: 10.1007/978-3-031-62743-9_1.
[85] K. N. Singh, N. Baranwal, O. P. Singh, and A. K. Singh, “SIELNet: 3-D Chaotic-Map-Based Secure Image Encryption Using Customized Residual Dense Spatial Network,” *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 862–868, Nov. 2023, doi: 10.1109/TCE.2022.3227401.
[86] Z. A. N. Fauzyah, A. Sambas, P. W. Adi, and D. R. I. M. Setiadi, “Quantum Key Distribution-Assisted Image Encryption Using 7D and 2D Hyperchaotic Systems,” *Journal of Future Artificial Intelligence and Technologies*, vol. 2, no. 1, pp. 47–62, Apr. 2025, doi: 10.62411/faith.3048-3719-93.
[87] V. Mamandi, N. Ardalani, and B. Ghalamkari, “A new attack resistant encryption method based on hybrid chaotic-quantum key distribution (CQKD),” *Quantum Inf Process*, vol. 23, no. 7, p. 265, Jul. 2024, doi: 10.1007/s11128-024-04434-6.
[88] Y. Hariprasad, S. S. Iyengar, and N. K. Chaudhary, “Securing the Future: Advanced Encryption for Quantum-Safe Video Transmission,” *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024, doi: 10.1109/TCE.2024.3473542.
[89] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” Mar. 14, 2020, *arXiv*: arXiv:2003.06557. doi: 10.48550/arXiv.2003.06557.
[90] C. Neuman, S. Hartman, K. Raeburn, and T. Yu, “The Kerberos Network Authentication Service (V5),” Internet Engineering Task Force, Request for Comments RFC 4120, Jul. 2005. doi: 10.17487/RFC4120.

[91] Kris Schirmer, “ba-arbeit-qkd-kerberos.” Accessed: Jul. 23, 2025. [Online]. Available: https://www.unibw.de/muquanet/publikationen-1/ba-arbeit-qkd-kerberos.pdf
[92] M. N. Wegman and J. L. Carter, “New hash functions and their use in authentication and set equality,” *Journal of Computer and System Sciences*, vol. 22, no. 3, pp. 265–279, Jun. 1981, doi: 10.1016/0022-0000(81)90033-7.
[93] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for Message Authentication,” Internet Engineering Task Force, Request for Comments RFC 2104, Feb. 1997. doi: 10.17487/RFC2104.
[94] M. J. Dworkin, “Recommendation for block cipher modes of operation : the CMAC mode for authentication,” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-38b, 2016. doi: 10.6028/NIST.SP.800-38b.
[95] Morris Dworkin, “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,” National Institute of Standards and Technology, NIST SP 800-38D, Nov. 2007. Accessed: Jul. 10, 2025. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
[96] E. O. Kiktenko *et al.*, “Lightweight Authentication for Quantum Key Distribution,” *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6354–6368, Oct. 2020, doi: 10.1109/TIT.2020.2989459.
[97] “WiN-Labor-Recherche_PQC-vs-QKD.pdf.” Accessed: Jul. 24, 2025. [Online]. Available: https://www.win-labor.dfn.de/files/2024/03/WiN-Labor-Recherche_PQC-vs-QKD.pdf
[98] National Institute of Standards and Technology (US), “Module-lattice-based key-encapsulation mechanism standard,” National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 203, Aug. 2024. doi: 10.6028/NIST.FIPS.203.
[99] National Institute of Standards and Technology (US), “Module-lattice-based digital signature standard,” National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 204, Aug. 2024. doi: 10.6028/NIST.FIPS.204.
[100] National Institute of Standards and Technology (US), “Stateless hash-based digital signature standard,” National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 205, Aug. 2024. doi: 10.6028/NIST.FIPS.205.
[101] “NIST Selects HQC as Fifth Algorithm for Post-Quantum Encryption,” *NIST*, Mar. 2025, Accessed: Jul. 10, 2025. [Online]. Available: https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption
[102] “Hamming Quasi-Cyclic (HQC) Fourth round version Updated version 19/02/2025”, Accessed: Jul. 11, 2025. [Online]. Available: https://pqc-hqc.org/doc/hqc-specification_2025-02-19.pdf
[103] M. Raavi, S. Wuthier, P. Chandramouli, X. Zhou, and S.-Y. Chang, “QUIC Protocol with Post-quantum Authentication,” in *Information Security*, W. Susilo, X. Chen, F. Guo, Y. Zhang, and R. Intan, Eds., Cham: Springer International Publishing, 2022, pp. 84–91. doi: 10.1007/978-3-031-22390-7_6.
[104] F. R. Ghashghaei, Y. Ahmed, N. Elmrabit, and M. Yousefi, “Enhancing the Security of Classical Communication with Post-Quantum Authenticated-Encryption Schemes for the Quantum Key Distribution,” *Computers*, vol. 13, no. 7, Art. no. 7, Jul. 2024, doi: 10.3390/computers13070163.
[105] D. Marchsreiter and J. Sepúlveda, “A PQC and QKD Hybridization for Quantum-Secure Communications,” in *2023 26th Euromicro Conference on Digital System Design (DSD)*, Sep. 2023, pp. 545–552. doi: 10.1109/DSD60849.2023.00081.
[106] L.-J. Wang *et al.*, “Experimental authentication of quantum key distribution with post-quantum cryptography,” *npj Quantum Inf*, vol. 7, no. 1, pp. 1–7, May 2021, doi: 10.1038/s41534-021-00400-7.
[107] Z. Zhang, Y. Li, and Z. Man, “Multiparty quantum secret sharing,” *Phys. Rev. A*, vol. 71, no. 4, Art. no. 4, Apr. 2005, doi: 10.1103/PhysRevA.71.044301.
[108] Q. Wang *et al.*, “A Novel Authenticated Quantum Anonymous Secret Sharing for Classical and Quantum Information,” *Advanced Quantum Technologies*, vol. n/a, no. n/a, p. 2400295, doi: 10.1002/qute.202400295.

[109] D. M. Greenberger, M. A. Horne, and A. Zeilinger, “Going Beyond Bell’s Theorem,” Dec. 06, 2007, *arXiv*: arXiv:0712.0921. doi: 10.48550/arXiv.0712.0921.
[110] Y.-H. Yang, F. Gao, X. Wu, S.-J. Qin, H.-J. Zuo, and Q.-Y. Wen, “Quantum secret sharing via local operations and classical communication,” *Sci Rep*, vol. 5, no. 1, Art. no. 1, Nov. 2015, doi: 10.1038/srep16967.
[111] Y. Long, C. Zhang, and Z. Sun, “Standard (3, 5)-threshold quantum secret sharing by maximally entangled 6-qubit states,” *Sci Rep*, vol. 11, no. 1, Art. no. 1, Nov. 2021, doi: 10.1038/s41598-021-01893-0.
[112] W. Dür, G. Vidal, and J. I. Cirac, “Three qubits can be entangled in two inequivalent ways,” *Phys. Rev. A*, vol. 62, no. 6, p. 062314, Nov. 2000, doi: 10.1103/PhysRevA.62.062314.
[113] G. Murta, F. Grasselli, H. Kampermann, and D. Bruß, “Quantum Conference Key Agreement: A Review,” *Adv Quantum Tech*, vol. 3, no. 11, p. 2000025, Nov. 2020, doi: 10.1002/qute.202000025.
[114] S. Liu, Z. Lu, P. Wang, Y. Tian, X. Wang, and Y. Li, “Experimental demonstration of multiparty quantum secret sharing and conference key agreement,” *npj Quantum Inf*, vol. 9, no. 1, Art. no. 1, Sep. 2023, doi: 10.1038/s41534-023-00763-z.
[115] A. Dutta and A. Pathak, “A short review on quantum identity authentication protocols: How would Bob know that he is talking with Alice?,” Dec. 08, 2021, *arXiv*: arXiv:2112.04234. doi: 10.48550/arXiv.2112.04234.
[116] A. Ghilen, H. Belmabrouk, and R. Bouallegue, “Classification of quantum authentication protocols and calculation of their complexity,” in *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, Dec. 2014, pp. 169–173. doi: 10.1109/STA.2014.7086669.
[117] Y. Yang, Q. Wen, and X. Zhang, “Multiparty simultaneous quantum identity authentication with secret sharing,” *Sci. China Ser. G-Phys. Mech. As*, vol. 51, no. 3, pp. 321–327, Mar. 2008, doi: 10.1007/s11433-008-0034-5.
[118] Z. Qu, X. Liu, and S. Wu, “Quantum identity authentication protocol based on three-photon quantum error avoidance code in edge computing,” *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 6, p. e3945, 2022, doi: 10.1002/ett.3945.
[119] K. Thapliyal, R. D. Sharma, and A. Pathak, “Orthogonal-state-based and semi-quantum protocols for quantum private comparison in noisy environment,” Aug. 03, 2017, *arXiv*: arXiv:1608.00101. doi: 10.48550/arXiv.1608.00101.
[120] M.-J. Geng, Y. Chen, T.-J. Xu, and T.-Y. Ye, “Single-state semiquantum private comparison based on Bell states,” Dec. 15, 2022, *arXiv*: arXiv:2111.13819. doi: 10.48550/arXiv.2111.13819.
[121] C. ho Hong, J. Heo, J. G. Jang, and D. Kwon, “Quantum identity authentication with single photon,” *Quantum Inf Process*, vol. 16, no. 10, p. 236, Aug. 2017, doi: 10.1007/s11128-017-1681-0.
[122] A. Babu and N. Shanthi, “Quantum Identity Authentication Using Non-Orthogonal State Encoding,” in *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, Ernakulam, India: IEEE, Sep. 2021, pp. 334–337. doi: 10.1109/ACCESS51619.2021.9563302.
[123] G. Chen *et al.*, “Quantum identity authentication protocol based on flexible quantum homomorphic encryption with qubit rotation,” *Journal of Applied Physics*, vol. 133, no. 6, p. 064402, Feb. 2023, doi: 10.1063/5.0135896.
[124] N.-H. Lim, J.-W. Choi, M.-S. Kang, H.-J. Yang, and S.-W. Han, “Quantum authentication method based on key-controlled maximally mixed quantum state encryption,” *EPJ Quantum Technol.*, vol. 10, no. 1, Art. no. 1, Dec. 2023, doi: 10.1140/epjqt/s40507-023-00193-y.
[125] V. Dunjko, P. Wallden, and E. Andersson, “Quantum Digital Signatures without Quantum Memory,” *Phys. Rev. Lett.*, vol. 112, no. 4, p. 040502, Jan. 2014, doi: 10.1103/PhysRevLett.112.040502.
[126] G. Roberts *et al.*, “Experimental measurement-device-independent quantum digital signatures,” *Nature Communications*, vol. 8, Oct. 2017, doi: 10.1038/s41467-017-01245-5.

[127] H.-L. Yin *et al.*, “Experimental quantum secure network with digital signatures and encryption,” *National Science Review*, vol. 10, no. 4, p. nwac228, Apr. 2023, doi: 10.1093/nsr/nwac228.
[128] M. Mosca, D. Stebila, and B. Ustaoglu, “Quantum Key Distribution in the Classical Authenticated Key Exchange Framework,” Jun. 26, 2012, *arXiv*: arXiv:1206.6150. Accessed: Jun. 11, 2024. [Online]. Available: http://arxiv.org/abs/1206.6150
[129] L. Garms *et al.*, “Experimental Integration of Quantum Key Distribution and Post-Quantum Cryptography in a Hybrid Quantum-Safe Cryptosystem,” *Advanced Quantum Technologies*, vol. 7, no. 4, p. 2300304, 2024, doi: 10.1002/qute.202300304.
[130] A. A. Giron, R. Custódio, and F. Rodríguez-Henríquez, “Post-quantum hybrid key exchange: a systematic mapping study,” *J Cryptogr Eng*, vol. 13, no. 1, pp. 71–88, Apr. 2023, doi: 10.1007/s13389-022-00288-9.
[131] E. Barker, A. Roginsky, and R. Davis, “Recommendation for cryptographic key generation,” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-133r2, Jun. 2020. doi: 10.6028/NIST.SP.800-133r2.
[132] E. Barker, L. Chen, and R. Davis, “Recommendation for Key-Derivation Methods in Key-Establishment Schemes,” National Institute of Standards and Technology, Aug. 2020. doi: 10.6028/NIST.SP.800-56Cr2.
[133] “T-TUT-ICTS-2022-1-PDF-E.pdf.” Accessed: Aug. 01, 2025. [Online]. Available: https://www.itu.int/dms_pub/itu-t/opb/tut/T-TUT-ICTS-2022-1-PDF-E.pdf
[134] S. Fluhrer, P. Kampanakis, D. McGrew, and V. Smyslov, “Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security,” RFC Editor, RFC8784, Jun. 2020. doi: 10.17487/RFC8784.
[135] V. Smyslov, “Intermediate Exchange in the Internet Key Exchange Protocol Version 2 (IKEv2),” Internet Engineering Task Force, Request for Comments RFC 9242, May 2022. doi: 10.17487/RFC9242.
[136] C. Tjhai *et al.*, “Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2),” Internet Engineering Task Force, Request for Comments RFC 9370, May 2023. doi: 10.17487/RFC9370.
[137] “TS 103 744 - V1.2.1 - CYBER; Quantum-Safe Cryptography (QSC); Quantum-safe Hybrid Key Establishment”.
[138] J. Blanco-Romero, P. O. García, D. Sobral-Blanco, F. A. Mendoza, A. F. Vilas, and M. Fernández-Veiga, “Hybrid Quantum Security for IPsec,” Jul. 12, 2025, *arXiv*: arXiv:2507.09288. doi: 10.48550/arXiv.2507.09288.
[139] S. Ranganathan, S. Karthikeyan, C. P. Chavan, and S. P. T, “Integrating Quantum Key Distribution (QKD) with Post-Quantum Cryptography (PQC): Combining BB84 Protocol with Lattice-Based Cryptographic Techniques,” in *2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG)*, Dec. 2024, pp. 1–9. doi: 10.1109/ICTBIG64922.2024.10911719.
[140] N. Aquina, S. Rommel, and I. T. Monroy, “Quantum secure communication using hybrid post-quantum cryptography and quantum key distribution,” in *2024 24th International Conference on Transparent Optical Networks (ICTON)*, Jul. 2024, pp. 1–4. doi: 10.1109/ICTON62926.2024.10648124.
[141] S. Ricci, P. Dobias, L. Malina, J. Hajny, and P. Jedlicka, “Hybrid Keys in Practice: Combining Classical, Quantum and Post-Quantum Cryptography,” *IEEE Access*, vol. 12, pp. 23206–23219, 2024, doi: 10.1109/ACCESS.2024.3364520.
[142] M. Geitz, R. Döring, and R.-P. Braun, “Hybrid QKD & PQC Protocols implemented in the Berlin OpenQKD testbed,” in *2023 8th International Conference on Frontiers of Signal Processing (ICFSP)*, Oct. 2023, pp. 69–74. doi: 10.1109/ICFSP59764.2023.10372894.
[143] P. Zeng *et al.*, “Practical hybrid PQC-QKD protocols with enhanced security and performance,” Nov. 07, 2024, *arXiv*: arXiv:2411.01086. doi: 10.48550/arXiv.2411.01086.
[144] B. Dowling, T. B. Hansen, and K. G. Paterson, “Many a Mickle Makes a Muckle: A Framework for Provably Quantum-Secure Hybrid Key Exchange,” in *Post-Quantum Cryptography*, J. Ding and J.-

P. Tillich, Eds., Cham: Springer International Publishing, 2020, pp. 483–502. doi: 10.1007/978-3-030-44223-1_26.
[145] S. Bruckner, S. Ramacher, and C. Striecks, “Muckle+: End-to-End Hybrid Authenticated Key Exchanges,” in *Post-Quantum Cryptography*, T. Johansson and D. Smith-Tone, Eds., Cham: Springer Nature Switzerland, 2023, pp. 601–633. doi: 10.1007/978-3-031-40003-2_22.
[146] C. Battarbee, C. Striecks, L. Perret, S. Ramacher, and K. Verhaeghe, “Quantum-Safe Hybrid Key Exchanges with KEM-Based Authentication,” Jul. 23, 2025, *arXiv*: arXiv:2411.04030. doi: 10.48550/arXiv.2411.04030.
[147] J. S. Buruaga, A. Bugler, J. P. Brito, V. Martin, and C. Striecks, “Versatile quantum-safe hybrid key exchange and its application to MACsec,” *EPJ Quantum Technol.*, vol. 12, no. 1, Art. no. 1, Dec. 2025, doi: 10.1140/epjqt/s40507-025-00382-x.
[148] “ETSI - xTF Portal.” Accessed: Sep. 26, 2025. [Online]. Available: https://portal.etsi.org/xtfs/#/xTF/684/
[149] A. Prakasan, K. Jain, and P. Krishnan, “Authenticated-Encryption in the Quantum Key Distribution Classical Channel Using Post-Quantum Cryptography,” in *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, May 2022, pp. 804–811. doi: 10.1109/ICICCS53718.2022.9788239.
[150] C. R. Garcia, A. C. Aguilera, J. J. V. Olmos, I. T. Monroy, and S. Rommel, “Quantum-Resistant TLS 1.3: A Hybrid Solution Combining Classical, Quantum and Post-Quantum Cryptography,” in *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Nov. 2023, pp. 246–251. doi: 10.1109/CAMAD59638.2023.10478407.