Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Telematik/Rechnernetze

Bachelorarbeit

Multipath Key Reinforcement in Weitverkehrsnetzen

vorgelegt von: Lukas Meinl
eingereicht am: 21. Oktober 2021
Studiengang: Informatik

Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. Günter Schäfer
Wissenschaftlicher Betreuer: Dr.-Ing. Michael Roßberg

URN: urn:nbn:de:gbv:ilm1-2021200215

DOI: 10.22032/dbt.58926

Zusammenfassung

Durch die drohende Existenz von praktikablen Quantencomputern steigt das Verlangen nach quantenresistenten Schlüsselaustauschprotokollen immer weiter. Vor allem in Weitverkehrsnetzen (WANs) ist das Absichern der Kommunikation entscheidend. Dabei setzt der momentane Stand der Wissenschaft auf die Post-Quanten-Kryptographie (PQC). Verfahren aus dieser Klasse haben aber entscheidende Nachteile: das Vertrauen in die Sicherheit fehlt oder die Performance ist im Vergleich zu etablierten Verfahren schlecht. Die Nachteile schließen sich dabei nicht gegenseitig aus. Aufgrund dessen wurde im Rahmen dieser Arbeit ein Schlüsselaustauschprotokoll entwickelt, welches auf Verfahren aus der PQC verzichtet. Das Verfahren bestimmt dabei möglichst viele Pfade, über die jeweils eine Zufallszahl ausgetauscht wird. Diese Zufallszahlen werden dann zu einem gemeinsamen Schlüssel abgeleitet. Die Erkundung der Wege ist dabei die zentrale Aufgabe des Protokolls, da sich die Sicherheit auf die Diversität und Anzahl der Pfade stützt, und wird mithilfe von begrenzten Fluten gelöst. Zusätzlich kann das Protokoll die erreichte Sicherheit automatisiert bewerten. Ein Quantencomputer stellt dabei keine große Bedrohung dar, da ein Angreifer alle Pfade kompromittieren muss, um den Schlüssel abzuleiten. Das Konzept wird dabei in ein VPN-Autokonfigurationsnetz integriert. Mithilfe einer Simulation wird dann das Konzept evaluiert, um die Eigenschaften Skalierbarkeit, Robustheit und Sicherheit zu untersuchen. Das entwickelte Konzept bietet dabei zusätzlich zu der Quantenresistenz hohe Robustheit gegenüber dynamischen Änderungen im Netz, logarithmische Abhängigkeit der Paketgröße von der Netzgröße und eine hohe Diversität der Pfade.

Abstract

Due to the threatened existence of practicable quantum computers, the demand for quantum-resistant key exchange protocols continues to grow. Securing communication is particularly important in wide area networks (WANs). The current state of science relies on post-quantum cryptography (PQC). However, algorithms from this class have significant disadvantages: there is no trust in security or the performance is poor compared to established algorithms. The disadvantages are not mutually exclusive. Because of this, a key exchange protocol was developed in this work, which does not use PQC algorithms. The method determines as many paths as possible, each of which is used to exchange a random number. These random numbers are then used to derive a key. Exploring paths is the central task of the protocol, since security relies on the diversity and number of paths and is solved with the help of limited flooding. In addition, the protocol can automatically evaluate the achieved security. A quantum computer does not represent a major threat because an attacker would have to compromise all paths in order to derive the key. The concept is integrated into a VPN auto-configuration network. With the help of a simulation the concept is then evaluated in order to examine the properties of scalability, robustness and security. In addition to the quantum resistance, the developed concept offers high robustness against dynamic changes in the network, logarithmic dependency of the packet size on the network size and a high diversity of the paths.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In der heutigen Zeit ist die Vernetzung der Welt einer der zentralsten Punkte unseres Lebens. Sie ermöglicht Privatpersonen von jedem beliebigen Punkt aus zu kommunizieren. Hinzu kommen immer mehr Smart-Gadgets für den Privatgebrauch, die dauerhaft über das Internet mit Anbietern von Dienstleistungen in Verbindung stehen. Doch auch für die Wirtschaft ist die Vernetzung essentiell. Egal ob Großkonzern oder Start-up – alle Unternehmen kommunizieren über das Internet.
Auch in der Zukunft wird die Vernetzung weiter fortschreiten. Die Digitalisierung wird weiter in den Alltag des Menschen vordringen. Der Großteil der Menschheit hinterfragt dabei nicht, wie dieses Wunder der Zeit funktioniert. Dabei birgt die Kommunikation über das Internet auch Risiken. Es muss sichergestellt werden, dass die Kommunikation nicht abgehört (Vertraulichkeit) und die Manipulation von Daten erkannt (Integrität) werden können. Für diese Aufgaben gibt es bereits eine Lösung: die symmetrische Kryptographie. Mithilfe von Verfahren wie AES [1] ist es möglich, Daten zu verschlüsseln und somit die Vertraulichkeit und Integrität zu sichern. Diese Verfahren benötigen aber ein vorher ausgehandeltes gemeinsames Geheimnis, wobei dieses Geheimnis die Grundlage der Sicherheit darstellt. Ist das Geheimnis unsicher, dann ist auch die Verschlüsselung der Daten unsicher. Bevor nun also die Daten verschlüsselt werden können, muss vorher ein Schlüsselaustausch stattfinden.
Das Problem des Schlüsselaustauschs über eine unsichere Verbindung galt lange als unlösbar. Doch in den 1970er Jahren wurde diese These gekippt und die ersten Public-Key-Kryptosysteme wurden veröffentlicht. Diese Kryptosysteme sind in der Lage über eine unsichere Verbindung einen Schlüssel auszuhandeln, ohne dass ein Angreifer das gemeinsame Geheimnis aus den öffentlichen Informationen ableiten kann. Die gängigen asymmetrischen Verfahren sind RSA [2] bzw. Verfahren aus der Klasse der Elliptische-Kurven-Kryptografie (ECC). Dabei haben die gängigen Verfahren eine Gemeinsamkeit. Deren Sicherheit beruht auf der vermuteten Komplexität zweier mathematischer Probleme: die Primfaktorzerlegung von großen Zahlen und die Berechnung des diskreten Logarithmus. Diese Probleme können mit gängigen Mitteln nicht effizient genug gelöst werden – die darauf basierenden Kryptosysteme sind also vermeintlich sicher.
Doch bereits 1994 wurde von Shor [3] gezeigt, dass diese Probleme effizient mithilfe von Quantencomputern lösbar sind. Dies hat zur Folge: Sobald kryptographisch relevante Quantencomputer existieren, sind die gängige asymmetrischen Kryptosysteme nicht mehr sicher. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) hat in der Studie *Entwicklungsstand Quantencomputer* [4] gezeigt, dass zwar kurzfristig nicht mit kryptographisch relevanten Quantencomputern zu rechnen ist. Da jedoch ein regelrechtes „Rennen“ um die Entwicklung solcher Quantencomputer herrscht und somit es nur noch eine Frage der Zeit ist, bis sie existieren, ist die Bedrohung real. Man benötigt nun neue Verfahren zum Schlüsselaustausch, die zusätzlich sicher gegenüber Quantencomputern sind. Die Post-Quanten-Kryptographie (PQC) bietet dabei bereits asymmetrische Lösungen, die theoretisch die Sicherheit gegenüber Quantencomputern garantieren. Praktisch haben diese Verfahren aber Nachteile: Entweder muss man mit Performance-Einbußen rechnen oder das Vertrauen in die Sicherheit ist noch nicht

gegeben [5, Kapitel 5].
In dieser Arbeit soll aufgrund dessen ein quantenresistentes Schlüsselaustauschprotokoll entwickelt werden, welches auf Verfahren zum Schlüsselaustausch aus der PQC verzichtet. Dabei liegt der Fokus auf einem Protokoll, welches in WANs eingesetzt werden soll. WANs sind Rechnernetze mit einer Vielzahl von Geräten, welche sich über einen sehr großen geographischen Bereich erstrecken. Beispiele dafür sind die Verbindung der Standorte eines Großkonzerns. Über diese Verbindungen werden sehr sensible Daten transportiert. Ein sicherer Schlüsselaustausch in diesen WANs ist dementsprechend fundamental.

1.1 Multipath Key Reinforcement

Die Grundidee für das zu entwickelnde Protokoll beruht auf der Idee des Multipath Key Reinforcements (MKRs) [6, Kapitel 11]. Die Grundlage für die Funktionalität dieses Verfahrens ist ein zusammenhängendes Netz. Die Verbindungen in diesem Netz müssen dabei sicher sein. Sicher in diesem Kontext bedeutet, dass solch eine Verbindung zwischen zwei Knoten mithilfe von symmetrischer Kryptographie abgesichert ist, was ein gemeinsames Geheimnis voraussetzt. MKR nutzt nun diese bestehenden Verbindung, um auf Grundlage deren Sicherheit einen neuen Schlüssel auszutauschen. Dafür werden zuerst mehrere, möglichst disjunkte Wege zwischen den beiden Kommunikationspartnern ermittelt. Über diese Wege werden dann verschiedene Zufallszahlen ausgetauscht. Der Schlüssel leitet sich dann aus diesen Zufallszahlen ab. Die Sicherheit dieses abgeleiteten Schlüssels ist dementsprechend abhängig von den gefundenen Wegen und deren Sicherheit.
MKR bietet dabei einen entscheidenden Vorteil: Das Verfahren ist quantenresistent, wenn die zugrunde liegenden Verbindungen nicht mithilfe eines Quantencomputers kompromittiert werden können. Denn um das abgeleitete Geheimnis zu berechnen, muss ein Angreifer alle Pfade kompromittieren, um an die ausgetauschten Zufallszahlen zu gelangen. Es gibt aber auch Nachteile bei diesem Verfahren. Eine Authentisierung des Austauschpartners ist mit MKR nicht möglich. Die Authentisierung wird aber in WANs benötigt, weswegen das Verfahren des MKRs dementsprechend erweitert werden sollte.

1.2 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in insgesamt sechs Kapitel. Im folgenden zweiten Kapitel werden zunächst wichtige mathematische Grundlagen und Begriffe, die für das weitere Verständnis essentiell sind, erläutert bzw. definiert.
Kapitel 3 beginnt zunächst mit der Formulierung der funktionalen, nichtfunktionalen und quantitativen Anforderungen an das zu entwickelnde Protokoll. Im darauffolgenden Abschnitt 3.4 werden wichtige Annahmen, die die Funktionalität des Protokolls sichern, formuliert. Anschließend wird das Angreifermodell, gegen welches das Protokoll sicher sein soll, beschrieben. Zuletzt wird der Stand der Technik beleuchtet, indem quantenresiste Schlüsselaustauschprotokolle beschrieben werden, die keine asymmetrische PQC benutzen.
In Kapitel 4 wird MuCKE, das zu entwickelnde quantenresistente Schlüsselaustauschprotokoll, vorgestellt. Die Funktionalität von MuCKE gliedert sich dabei in vier Abschnitte. Im ersten Abschnitt wird das Erkunden der Wege erläutert. Darauffolgend werden die Schlüsselableitung beschrieben und wichtige Mechanismen wie das Updaten des Schlüssels diskutiert. Im Abschnitt 4.3 werden die automatisierte Sicherheitsbewertung

beschrieben und deren Einschränkungen erläutert. Im letzten Abschnitt werden Details der Implementierung aufgezeigt und erläutert.
Das vorletzte Kapitel 5 beschäftigt sich mit dem Nachweis erreichter Anforderungen und Ziele. Dazu wird im ersten Abschnitt die Umsetzung der qualitativen Anforderungen diskutiert. Im nachfolgenden Abschnitt werden weitere quantitative Anforderungen mithilfe einer Simulation evaluiert. Dazu wird das Konzept in das VPN-Autokonfigurationssystem SOLID integriert.
Diese Arbeit schließt mit Kapitel 6 ab, welches aus einer kurzen Zusammenfassung und einem Ausblick auf fortsetzende Arbeiten an dem Protokoll besteht.

Kapitel 2

Grundlagen

In diesem Kapitel wird zuerst der Begriff des Schlüsselaustauschs erläutert. Darauffolgend werden weitere Begriffe, die mit dem Thema des Schlüsselaustauschs zusammenhängen, beschrieben. Im Abschnitt 2.4 wird SOLID vorgestellt, welches die Funktion der VPN-Autokonfiguration realisiert und somit ein Rechnernetz verwaltet. Zuletzt werden wichtige mathematische Begriffe und Notationen, die für das weitere Verständnis essenziell sind, definiert.

2.1 Schlüsselaustausch

Ein Schlüsselaustausch ist ein kryptographisches Verfahren, welches zwischen zwei Kommunikationspartnern ein gemeinsames Geheimnis austauscht. Der Schlüsselaustausch hat dabei das Ziel, dass danach nur die beiden Kommunikationspartner das Geheimnis kennen. Das grundlegende Problem: Wie tauscht man sicher ein Geheimnis über einen unsicheren Kanal aus? Man könnte das Geheimnis mit einem Schlüssel verschlüsseln, doch wie tauscht man dann diesen Schlüssel aus? Dieses Paradoxon galt lange als unlösbar. Die einzige Möglichkeit war für lange Zeit das manuelle Verteilen von sogenannten Pre-shared Keys (PSKs). Dies ist sehr aufwendig und die Sicherheit beruht auf dem Vertrauen der Kuriere, die diese PSKs verteilen. Erst in den 1970er Jahren wurde durch die Erfindung asymmetrischer Verfahren das Problem des Schlüsselaustauschs über einen unsicheren Kanal gelöst.
In der heutigen Zeit steht man erneut vor diesem Problem, denn die gängigen Verfahren werden, sobald kryptographisch relevanten Quantencomputer existieren, nicht mehr sicher sein.

2.1.1 Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) ist eine Eigenschaft von Protokollen bzw. Verfahren. Erfüllt ein Verfahren die Eigenschaft der PFS, dann soll es einem Angreifer nicht möglich sein, eine vergangene verschlüsselte Kommunikation zu entschlüsseln, falls er das gemeinsame Geheimnis (auch Langzeitschlüssel genannt) kennt.
Dies wird durch sogenannte Sitzungsschlüssel erreicht. Wollen also Alice und Bob wieder kommunizieren, so wird eine Sitzung initiiert und dabei ein neuer Schlüssel ausgehandelt. Dieser Sitzungsschlüssel wird mit dem Langzeitschlüssel kombiniert und ergibt einen neuen Schlüssel, mit dem die Kommunikation der aktuellen Sitzung verschlüsselt werden kann.

2.2 Post-Quanten-Kryptographie

Die Post-Quanten-Kryptographie (PQC) ist ein Teilgebiet der Kryptographie und umfasst alle kryptographischen Verfahren, die gegenüber Quantencomputern resistent sind. Mithilfe der Schlüsselaustauschverfahren aus der PQC lässt sich die PFS realisieren. In diesem Abschnitt sollen drei ausgewählte Klassen der PQC anhand eines Verfahrens

vorgestellt werden. Die drei vorzustellenden Klassen sind dabei die hashbasierte, die codebasierte und die gitterbasierte Kryptographie. Die Grundlage für diesen Überblick stammt dabei aus [5].

2.2.1 Hashbasierte Kryptographie

Die Klasse der hashbasierten Kryptographie bietet vielversprechende Signaturverfahren. Eines dieser Verfahren ist das eXtended Merkle Signature Scheme (XMSS) [7]. Dieses Verfahren kombiniert gute Laufzeit mit einer praktikablen Größe der Schlüssel (öffentliche Schlüssel sind ca. 1700 B groß [5, S.9]) und kann dadurch mit gängigen Signaturverfahren wie RSA konkurrieren. Dazu kommt der enorme Vorteil der Quantenresistenz. Der Nachteil von XMSS: Dieses Verfahren ist zustandsbehaftet. Das hat zur Folge, dass nur endliche viele Signaturen mit einem öffentlichen Schlüssel erzeugt werden können. Je nach Wahl der Parameter sind bis zu 2^{20}-viele Signaturen mit einem öffentlichen Schlüssel möglich.

2.2.2 Codebasierte Kryptographie

Eines der Vertreter dieser Klasse ist das McEliece-Kryptosystem [8], welches bereits in den 1970er Jahren entwickelt und seitdem stetig weiter verbessert worden ist. Der Vorteil dieses Verfahrens ist das hohe Vertrauen in die Sicherheit. In den mittlerweile fast 50 Jahren Forschung an diesem Protokoll ist bis heute keine Sicherheitslücke oder Angriffsmuster entdeckt worden, um das Verfahren zu kompromittieren. Der Nachteil dieses Verfahrens ist die enorme Größe des öffentlichen Schlüssels. Je nach vorgeschlagenem Standard werden die Public-Keys zwischen 1 bis 1,3 MB groß [9, S. 31-32]. Diese Public-Keys sind also um den Faktor ≈ 2500 größer als die Public-Keys von gängingen RSA-Standards (RSA-4096 mit einem 512 B großen Public-Key).

2.2.3 Gitterbasierte Kryptographie

Der bekannteste Vertreter aus dieser Klasse ist das NTRU-Kryptosystem [10], welches erst in den 1990er Jahren entwickelt worden ist. Im Gegensatz zum McEliece-Kryptosystem hat dieses Verfahren kein Problem mit der Effizienz hinsichtlich der Schlüsselgrößen. Der Nachteil von NTRU (und generell gitterbasierten Verfahren) ist, dass dieses Verfahren erst seit ca. 20 Jahren existiert und somit die Kryptanalyse noch nicht weit fortgeschritten ist, wie dies bei RSA oder McEliece der Fall ist. Das Vertrauen in die Sicherheit ist u.a. aus diesem Grund zum jetzigen Zeitpunkt noch nicht gegeben.

2.3 Quantenschlüsselaustausch

Der Quantenschlüsselaustausch (QKD) [11] beschreibt eine Klasse von Verfahren, die mithilfe der Quantenmechanik eine Zufallszahl zwischen zwei Kommunikationspartnern austauscht. Dazu wird ein Quantenkanal zwischen diesen beiden Kommunikationspartnern benötigt. Neben diesem Quantenkanal existiert meistens noch ein zweiter, klassischer Kanal, über den die Kommunikation geleitet wird. Über den Quantenkanal kann dann ein Schlüssel ausgetauscht werden, mit dessen Hilfe man die Kommunikation über den zweiten Kanal absichern kann.
Die Sicherheit von einem QKD-Verfahren beruht dabei nicht auf vermuteten Annahmen der Komplexität von mathematischen Problemen, wie es bei der klassischen und der Post-Quanten-Kryptographie der Fall ist, sondern auf den Gesetzen der Quantenmechanik. QKD-Verfahren erkennen dabei (beim Überschreiten eines bestimmten

Fehler-Toleranzwertes) ein Abhören des Quantenkanals und können dadurch den Schlüsselaustausch sofort beenden.
Des Weiteren kann mithilfe von QKD-Verfahren PFS realisiert werden. Dazu wird zuerst zwischen den beiden Kommunikationspartnern ein PSK verteilt. Dieser wird zur Authentisierung des Quantenkanals genutzt. Danach kann mithilfe des Quantenkanals für jede neue Sitzung ein neuer Schlüssel ausgetauscht werden.
Die Nachteile von QKD-Verfahren sind die enormen Kosten und technischen Hürden, um solch einen Quantenkanal zu realisieren.

2.4 Secure OverLay for IPsec Discovery (SOLID)

SOLID [12] ist ein System zur VPN-Autokonfiguration und verwaltet dementsprechend ein VPN-Rechnernetz. Das zu entwickelnde Schlüsselaustauschprotokoll soll in das SOLID-System eingepflegt werden. Aus diesem Grund werden in diesem Abschnitt wichtige Funktionalitäten von SOLID erläutert, die das zu entwickelnde Protokoll später nutzen wird.

2.4.1 Netzstruktur

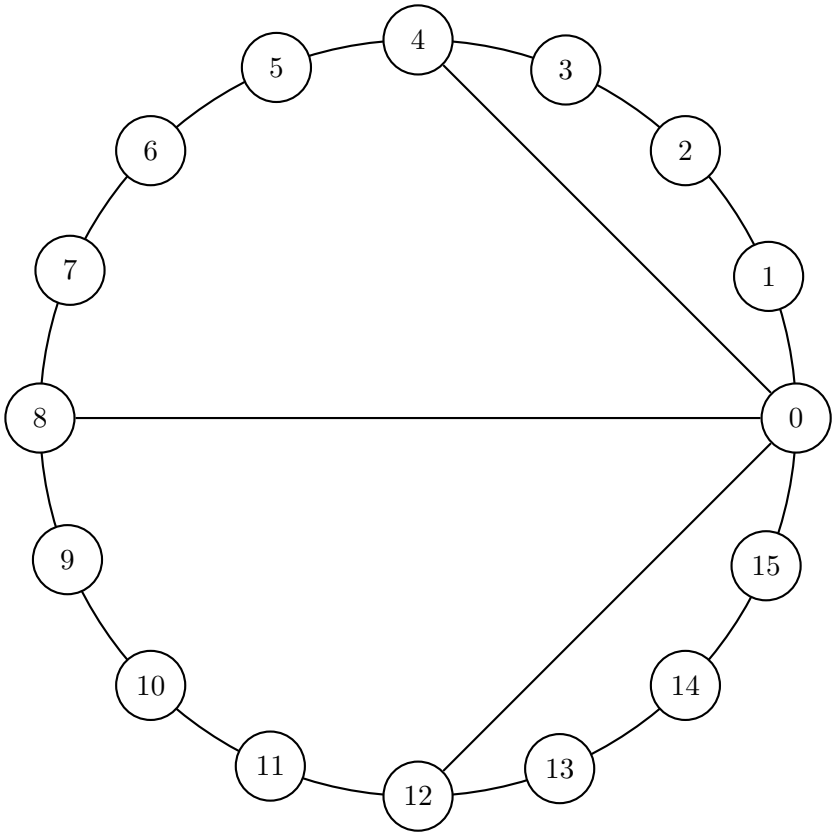


Abbildung 2.1: Ein Beispiel für eine Ringstruktur eines Netzes, indem der Knoten 0 bereits Querverbindungen aufgebaut hat.

Die Netzstruktur von SOLID ist abhängig davon, in welcher Schicht man diese betrachtet. Für diese Arbeit ist die Struktur des VPN-Overlays ausreichend, um die genutzten Funktionalitäten von SOLID zu erläutern. Die Netzstruktur solch eines Overlays basiert

bei SOLID auf einer Ringstruktur, wie sie in der Abbildung 2.1 abgebildet ist. Dabei wird angenommen, dass die einzelnen Knoten eine ID besitzen. Jeder Knoten hat dabei eine Verbindung zu seinem (von der ID abhängigen) Nachbarn. Zusätzlich zu diesen direkten Verbindungen baut das Netz Querverbindungen auf, um die Suche nach Netzknoten auf $\mathcal{O}(\log n)$, wobei n die Anzahl der Knoten bezeichnet, zu beschränken. In der Abbildung 2.1 hat der Knoten 0 also bereits Querverbindungen zum Knoten 4, 8 und 12 aufgebaut. Die Verbindungen sind dabei bidirektional.

2.4.2 Routing von Paketen

Lässt man die Verteilung der Netzlast außen vor, dann ist das Routing in solchen Ringstrukturen simpel. Die Netzlastverteilung ist für das Routen von einzelnen Kontrollnachrichten von Protokollen nicht relevant. Möchte also der Knoten 0 zum Knoten 11 ein Paket versenden, so betrachtet er seine Nachbarn und berechnet jeweils die Differenzen von der ID des Nachbarn und der ID des Zielknotens. Das Paket wird zu dem Knoten geroutet, bei welchem diese Differenz minimal ist. In dem Beispiel wird das Paket also zum Knoten 12 geroutet.

2.5 Mathematische Begriffe und Notationen

Im Folgenden werden nun Notationen und Begriffe definiert, die in dieser Arbeit verwendet werden.

Definition 1 (Notationen)**.** *Folgende Notationen werden in dieser Arbeit verwendet:*

- *Die Konkatenation zweier Bitfolgen wird mit $\|$ bezeichnet.*
- *Mengen, deren Elemente selbst wiederum Mengen sind, werden mit einem Großbuchstaben in Frakturschrift (z.B. $\mathfrak{S}$) bezeichnet.*
- *Die Potenzmenge wird mit $\mathfrak{P}$ bezeichnet.*
- *Eine Kante $\{x, y\}$ eines ungerichteten Graphen $G = (V, E)$ wird oftmals mit (x, y) bezeichnet. Es gilt also für alle Kanten eines Graphen: $(x, y) = (y, x)$.*

Wird im Kontext dieser Arbeit von Netzen gesprochen, so wird damit immer ein dazugehöriger ungerichteter Graph $G = (V, E)$ beschrieben. Dabei entspricht die Menge der Knoten V gerade den Geräten im Netz (Netzknoten). Die Menge der Kanten E entspricht dabei den bestehenden Verbindungen zwischen diesen Knoten. Da es sich um einen ungerichteten Graphen handelt, können zwei Knoten immer in beide Richtungen kommunizieren. Verbindungen, bei denen die Kommunikation nur in eine Richtung erlaubt ist, sind ausgeschlossen. Ein wichtiger Begriff in diesem Kontext ist die Nachbarschaft eines Knotens:

Definition 2 (Nachbarschaft)**.** *Sei $G = (V, E)$ ein ungerichteter Graph. Dann heißen zwei Knoten $u, v \in V$ benachbart (oder auch verbunden, adjazent) in G, falls sie durch eine Kante verbunden sind, also $(u, v) \in E$ gilt. Die Menge aller Nachbarn eines Knotens $v \in V$ in G wird mit $N_G(v)$ bezeichnet.*

Aufgrund der Klarheit über welches Netz (und somit über welchen Graphen) gerade Aussagen getroffen werden, wird oftmals $N(v)$ anstatt $N_G(v)$ geschrieben. Eine Besonderheit der Netze, die in dieser Arbeit betrachtet werden, sind die sogenannten Sicherheitswahrscheinlichkeiten von Knoten und Kanten:

Definition 3 (Sicherheitswahrscheinlichkeit)**.** *Sei $G = (V, E)$ ein ungerichteter Graph. Jedes Element $x \in V \cup E$ besitzt eine Sicherheitswahrscheinlichkeit p_x, welche die Wahrscheinlichkeit angibt, dass der Knoten bzw. die Kante x nicht kompromittiert ist.*

Da mithilfe von MKR ein Schlüsselaustausch vollzogen werden soll, ist der Begriff des Pfades essenziell für das weitere Verständnis:

Definition 4 (Pfad)**.** *Sei $G = (V, E)$ ein ungerichteter Graph. Ein Teilgraph $W = (V', E')$, wobei*

- $\{v_1, v_2, \ldots, v_n\} = V' \subseteq V$
- $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\} = E' \subseteq E$

mit $n \geq 2$ gilt, heißt Pfad (oder auch Weg), wenn alle Knoten $v_i, i \in \{1, \ldots, n\}$ paarweise disjunkt sind. Auch ein Teilgraph $W = (\{v\}, \emptyset)$ für ein $v \in V$ heißt Pfad (oder auch Weg).

Im Abschnitt 4.3 wird die automatisierte Bewertung der erreichten Sicherheit beschrieben. Dabei ist die Berechnung sogenannter Schwachstellen des Netzes, die mithilfe von Äquivalenzklassen realisiert wird, ausschlaggebend.

Definition 5 (Schwachstelle)**.** *Sei $G = (V, E)$ ein ungerichteter Graph und $\mathfrak{W} = \{(V_1, E_1), \ldots, (V_n, E_n)\}$ eine Menge von Pfaden aus G. Ein Teilgraph $S = (V', E')$ heißt Schwachstelle, falls ein $J \subseteq \{1, \ldots, n\}$ mit $|J| \geq 2$ existiert mit*

$$\forall v \in V', j \in J : v \in V_j \text{ und}$$
$$\forall e \in E', j \in J : e \in E_j.$$

Definition 6 (Äquivalenzrelation)**.** *Sei $A \neq \emptyset$ eine beliebige Menge. Eine Relation $\sim \subseteq A \times A$ heißt Äquivalenzrelation, falls für alle $x, y, z \in A$ gilt:*

- *$x \sim x$ (Reflexivität)*
- *$x \sim y \Rightarrow y \sim x$ (Symmetrie)*
- *$x \sim y$ und $y \sim z \Rightarrow x \sim z$ (Transitivität)*

Definition 7 (Äquivalenzklassen)**.** *Sei $\sim$ eine Äquivalenzrelation auf einer Menge $A \neq \emptyset$. Die Teilmenge $[a]_\sim := \{b \in A \mid b \sim a\}$ ist die Menge aller zu $a \in A$ äquivalenten Elemente und heißt Äquivalenzklasse von a.*

Wenn in dieser Arbeit Äquivalenzklassen gebildet werden, dann ist die zugrunde liegende Äquivalenzrelation aus dem Kontext klar erschließbar. Aufgrund dessen wird dann $[a]$ anstatt $[a]_\sim$ geschrieben.

Kapitel 3

Anforderungen an quantenresistente Schlüsselaustauschprotokolle

In diesem Kapitel werden zunächst die Anforderungen an das zu entwickelnde Protokoll strukturiert dargestellt, um darauffolgend den Stand der Technik zu analysieren.

3.1 Funktionale Anforderungen

Das Protokoll soll folgende Anforderungen erfüllen:

1. **Schlüsselaustausch:** Die zentrale Aufgabe des Protokolls ist die Ableitung eines gemeinsamen Geheimnisses zwischen zwei Netzwerkknoten in einem WAN.
2. **Sicherheitsbewertung:** Nach einem Schlüsselaustausch soll das Protokoll die Sicherheit des gemeinsamen Geheimnisses bewerten können. Diese Sicherheit soll mithilfe abgeschätzter Sicherheitswahrscheinlichkeiten der Knoten bzw. Kanten abgeleitet und in Prozent angegeben werden.
3. **Perfect Forward Secrecy:** Das Protokoll soll durch erneutes Ausführen den Schlüssel zwischen zwei Netzknoten erneuern. Dabei soll das Kompromittieren vorheriger Schlüssel die Sicherheit des neuen abgeleiteten Schlüssel nicht beeinflussen.

3.2 Nichtfunktionale Anforderungen

Außerdem soll das Protokoll weitere Anforderungen erfüllen, welche die Qualität des Ansatzes erfassen. Diese sind:

1. **Skalierbarkeit:** Das Protokoll soll unabhängig von der Größe des Netzes, d.h. der Anzahl der Knoten und Kanten, seine Eigenschaften erfüllen.
2. **Sicherheit:** Die zentrale Sicherheitseigenschaft des Protokolls lässt sich auf die Sicherheit des ausgehandelten gemeinsamen Geheimnisses reduzieren. Das gemeinsame Geheimnis gilt demnach als unsicher bzw. kompromittiert, falls eine dritte, nicht-autorisierte Instanz das gemeinsame Geheimnis kennt.
3. **Robustheit:** Aufgrund der Eigenschaft von WANs soll das Protokoll mit dynamischen Ereignissen, d.h. hinzukommende oder wegfallende Knoten bzw. Kanten, umgehen können.

3.3 Quantitative Anforderungen

Folgende quantitative Anforderungen soll das Protokoll erfüllen:

1. **Laufzeit:** Die Schlüsselaushandlung soll nicht länger als 3 s dauern.
2. **Paketgröße:** Die einzelnen Pakete sollen nicht größer als 100 B sein.

3. **Netzlast:** Die Anzahl der Pakete soll nicht größer als 50 Pakete pro Schlüsselaushandlung sein.

3.4 Annahmen

Damit die Funktionalität des Protokolls gewährleistet werden kann, müssen folgende Annahmen gelten:

1. **Zusammenhängend:** Das gegebene Netz muss zusammenhängend sein. Das bedeutet, je zwei Knoten aus dem Netz können entweder direkt oder über mehrere Hops kommunizieren. Außerdem kennt jeder Knoten seine direkten Nachbarn.
2. **Hop-zu-Hop Verschlüsselung:** Die direkten Verbindungen im Netz müssen mittels symmetrischer Verschlüsselung gesichert sein.
3. **Sicherheitswahrscheinlichkeiten:** Damit das Protokoll die Sicherheit bewerten kann, müssen die einzelnen Knoten und Kanten eine projizierte Sicherheitswahrscheinlichkeit besitzen. Die Sicherheitswahrscheinlichkeit der Knoten kann mithilfe ihrer Zertifikate bestimmt werden. Die Sicherheitswahrscheinlichkeit einer Kante ist ihren Knoten bekannt.

3.5 Angreifermodell

In diesem Abschnitt wird das Angreifermodell, gegen welches das Protokoll sicher sein soll, beschrieben. Dazu betrachte man zuerst das Dolev-Yao-Modell [13]. In diesem formalen Modell existiert ein aktiver Angreifer mit folgenden Eigenschaften:

- Der Angreifer ist ein berechtigter Teilnehmer des Netzes und kann Nachrichten an jeden anderen Teilnehmer senden und von jedem anderen Teilnehmer empfangen.
- Der Angreifer kann jede Nachricht abfangen.
- Der Angreifer kann seine Nachrichten unter einer falschen Identität versenden.

Zusätzlich zu dem Dolev-Yao-Modell verfügt der Angreifer über einen Quantencomputer, mit dem er gängige asymmetrische Verfahren brechen kann.

3.6 Stand der Technik

In diesem Abschnitt werden existierende quantenresistente Schlüsselaustauschprotokolle vorgestellt und analysiert. Dazu wird zuerst begründet, warum asymmetrische Verfahren aus der PQC keine Alternativen sind. Darauffolgend werden quantenresistente Schlüsselaustauschprotokolle beschrieben, die auf Verfahren aus der PQC verzichten.

3.6.1 Post-Quanten-Kryptographie

Die PQC wurde bereits im Abschnitt 2.2 beschrieben. In diesem Abschnitt soll die in Kapitel 1 getroffene Aussage, dass asymmetrische Verfahren aus der PQC entweder eine schlechte Performance haben oder das Vertrauen in die Sicherheit noch fehlt, begründet werden. Die Grundlage für diese Diskussion stammt dabei aus [5]. Die Klasse der hashbasierten Kryptographie bietet eine gute Alternative, um quantenresistente Signaturen zu erzeugen. Sie bietet aber keine Lösungen für Verfahren zum Schlüsselaustausch. Die codebasierte Kryptographie hat den Vorteil der hohen Sicherheit, welche

aber mit Performance-Einbußen einher geht. Aus diesem Grund sind Verfahren aus dieser Klasse keine Alternative für die genannten Anforderungen. Zuletzt gibt es noch die gitterbasierte Kryptographie. Verfahren aus dieser Klasse bieten gute Performance. Der große Nachteil bei dieser Klasse ist, dass die Verfahren aus dieser Klasse noch sehr jung sind und deswegen die Kryptoanalyse noch nicht sehr weit fortgeschritten ist. Das Vertrauen in die Sicherheit ist aus diesem Grund noch nicht gegeben.
Asymmetrische Verfahren zum Schlüsselaustausch aus der PQC bieten zwar eine quantenresistene Lösung, welche aber immer mit Nachteilen einhergehen, die mit den gestellten Anforderungen im Widerspruch stehen. Aus diesem Grund sind solche Verfahren in diesem Kontext keine Lösungen. Hashbasierte Signaturen bieten hingegen eine Möglichkeit quantenresistente Authentisierungen zu vollziehen.

3.6.2 Pre-shared Keys

PSKs sind per Definition quantenresistent. Ein Angreifer müsste das Verteilen der Schlüssel, welches mithilfe von Kurieren gelöst wird, abhören. Ein Quantencomputer bietet dem Angreifer deswegen keinen Vorteil. PSKs haben dafür aber enorme Nachteile. Das einmalige Aushandeln eines Schlüssels ist bereits sehr aufwendig und kostenintensiv. Versucht man mithilfe von PSKs die PFS zu erfüllen, dann ist dies nicht realisierbar. Man überlege sich folgendes Szenario: Eine Firma hat zwei Standorte. Der Hauptsitz ist in Berlin, der Zweitsitz in London. Wollen diese beiden nun sicher kommunizieren und dabei die PFS erfüllen, dann müsste jede Stunde ein Kurier mit einem Flugzeug von Berlin nach London fliegen, um dann jeweils für eine Stunde eine Sitzung aufzubauen. Die Kosten für die Flüge und die Boten im Zusammenhang mit der geringen Sicherheit, da mehrere Personen an dem Schlüsselaustausch beteiligt sind, machen dieses Verfahren unbrauchbar. Die PFS mithilfe von PSKs zu erfüllen ist also undenkbar aufwendig. Aus diesem Grund ist das Verteilen von PSKs mithilfe von Kurieren keine Alternative.

3.6.3 IKEv2 RFC 8784

Der momentane IKEv2-Standard, der im RFC 7296 [14] spezifiziert ist, nutzt noch gängige asymmetrische Verfahren und bietet deshalb keinen Schutz vor Quantencomputern. Zum jetzigen Stand gibt es bereits einen Entwurf für ein Framework [15], mithilfe dessen man quantenresistente Schlüsselaustauschprotokolle mit IKEv2 verbinden kann. Einen quantenresistenten IKEv2-Standard, der auf Verfahren aus der PQC zurückgreift, gibt es zum jetzigen Zeitpunkt noch nicht.
Eine quantenresistente Alternative, die ohne asymmetrische PQC auskommt, bietet der RFC 8784 [16]. Die Grundidee ist dabei, die Quantenresistenz mithilfe von PSKs zu garantieren. Das Verfahren nimmt dazu an, dass zwischen den beiden Schlüsselaustauschpartnern ein PSK existiert. Dieser PSK wird dann in alle Schlüsselableitungen miteinbezogen um die Quantenresistenz zu sichern. Das Problem dabei: Das Verteilen der PSKs ist nicht weiter spezifiziert. Selbst wenn sich die beiden Partner bereits in einem zusammenhängenden Netz befinden, muss vor einem Schlüsselaustausch ein PSK zwischen diesen beiden Partnern verteilt werden. Solch ein Protokoll skaliert dementsprechend schlecht in WANs, was aber eine Anforderung an das zu entwickelnde Protokoll ist.

3.6.4 Multipath Key Reinforcement in Wireless Sensor Networks

Das Verfahren des MKRs [6, Kapitel 11] stammt aus dem Kontext der Wireless Sensor Networks (WSNs). Ein WSN ist ein Rechnernetz, wobei die Knoten Sensoren darstellen,

welche untereinander drahtlos kommunizieren. Die Herausforderung dabei ist, dass die Sensoren keine große Speicher- und Rechenkapazität haben. Asymmetrische Verfahren zum Schlüsselaustausch sind deswegen zu aufwendig und können nicht von diesen Sensoren ausgeführt werden. In dieser Art von Netzen müssen also Schlüssel ausgetauscht werden mit dem Verzicht auf asymmetrische Kryptographie. Diese Anforderung hat dabei Ähnlichkeiten zu den in diesem Kapitel genannten Anforderungen. Der Unterschied ist, dass in diesem Szenario auf asymmetrische Verfahren aus der PQC verzichtet wird, wohingegen in WSNs generell asymmetrische Verfahren keine Alternative darstellen.
Um dieses Problem in WSNs zu lösen, wurde MKR entwickelt. Die Grundidee wurde bereits im Abschnitt 1.1 beschrieben: Man nutzt mehrere Pfade, über die man Zufallszahlen austauscht. Dadurch kann auf Basis der bestehenden Verbindungen die Sicherheit verstärkt werden. Die Annahme dabei ist, dass das gegebene Netz zusammenhängend ist.
Da die Eigenschaft des zusammenhängenden Netzes in dem in dieser Arbeit betrachteten Szenario gegeben ist (siehe Abschnitt 3.4), soll mithilfe von MKR ein Schlüsselaustausch in WANs realisiert werden. Dabei gibt es aber auch entscheidende Unterschiede von WANs zu WSNs:

- Die Netzknoten in WANs haben wesentlich mehr Speicher- und Rechenkapazität.
- Die Latenz zwischen den einzelnen Netzknoten in WANs ist wesentlich höher.

Darauf muss bei der Realisierung geachtet werden, um die in diesem Kapitel gestellten Anforderungen nicht zu verletzen.

Zusammenfassung

Der momentane Stand der Technik setzt bei quantenresistenten Schlüsselaustauschprotokollen vor allem auf Verfahren aus der PQC. Schlüsselaustauschprotokolle für WANs, die auf Verfahren aus der PQC verzichten, sind selten. Der RFC 8784 bietet zwar eine Alternative, die aber auf PSKs beruht und somit nicht gut skaliert. Einzig und allein das MKR, welches aus dem Kontext der WSNs entstammt, bietet eine solide Grundlage um ein quantenresistentes Schlüsselaustauschprotokoll ohne PQC zu entwickeln.

Kapitel 4

Multipath Cryptographic Key Exchange (MuCKE)

Aufgrund der im letzten Kapitel beschriebenen Defizite bei quantenresistenten Schlüsselaustauschprotokollen wurde im Rahmen dieser Arbeit ein neues Protokoll mit dem Namen Multipath Cryptographic Key Exchange (MuCKE) entwickelt. In diesem Kapitel wird das Konzept von MuCKE beschrieben und die dabei getroffenen Entwurfsentscheidungen werden begründet. Die Aufgabe von MuCKE ist es, ein gemeinsames Geheimnis zwischen zwei Netzknoten (im Folgenden als Alice und Bob bezeichnet) abzuleiten. Dafür wird im ersten Schritt ein Multipath-Schlüsselaustausch ausgeführt. Die Idee dieses Schlüsselaustauschs beruht auf der Methode des MKRs. Darauffolgend wird ein IKEv2-Schlüsselaustausch [14] ausgeführt. MuCKE erweitert also den IKEv2-Standard mithilfe des Multipath-Schlüsselaustauschs.
Die Aufgaben des Protokolls lassen sich logisch in drei Teile aufteilen: das Erkunden der Wege (Abschnitt 4.1), das Ableiten des Schlüssels (Abschnitt 4.2) und die automatisierte Bewertung der erreichten Sicherheit (Abschnitt 4.3). Die Lösungen dieser Aufgaben werden in den folgenden Abschnitten beschrieben und die dabei getroffenen Entwurfsentscheidungen begründet.

4.1 Erkundung der Wege

Zuerst hat das Protokoll die Aufgabe mehrere Wege zwischen Alice und Bob zu finden. Eine erster, einfacher Ansatz wäre eine zentrale Lösung: Ein zentraler Server kennt die aktuelle Netztopologie. Durch diese Informationen kann er mehrere möglichst disjunkte Wege zwischen zwei beliebigen Netzknoten finden. Alice müsste dementsprechend dem Server eine Anfrage stellen und würde dann als Antwort mehrere Wege bekommen. Das große Problem solcher Ansätze ist, dass dieser zentrale Server ein Single Point of Failure (SPOF) ist. SPOFs stellen ein Sicherheitsrisiko dar und sind zusätzlich nicht besonders robust.
Aufgrund dieses Problems bei zentralen Ansätzen wird in dieser Arbeit ein verteilter Ansatz verfolgt. Aber auch nicht jeder verteilte Ansatz eignet sich für die gegebenen Anforderungen. Eine Möglichkeit wäre es, den Algorithmus von Dijkstra [17] zu nutzen. Mit ihm kann der kürzeste Weg zwischen zwei Knoten in einem Graphen berechnet werden. Diese Eigenschaft könnte man nutzen um mehrere Wege zwischen Alice und Bob zu finden. Dazu benötigt man aber aktuelle Information über die Netztopologie, die davor gesammelt werden müssten. Das Problem bei diesem Ansatz: WANs sind in der Regel dynamisch, d.h. Knoten und Kanten können jederzeit wegfallen oder hinzukommen. Solch eine Änderung der Netztopologie muss erkannt werden, damit der Dijkstra-Algorithmus auf der aktuellen Netztopologie ausgeführt werden kann. Dazu müsste man vor jedem Schlüsselaustausch die Informationen der Netztopologie erneuern. Da dies aber häufig der Fall ist, skaliert dieser Ansatz schlecht mit der Größe des Netzes, was im Widerspruch zum Punkt 1 der nichtfunktionalen Anforderungen steht.
Eine weitere Idee ist das Erkunden der Wege durch Fluten zu lösen. Alice flutet sogenannte Erkundungspakete (EPs), auf die Bob antwortet. Doch das Fluten von Paketen in WANs skaliert wiederum schlecht. Es gibt aber auch Vorteile des Ansatzes:

Die Dynamik von WANs beeinflusst diese Idee nicht negativ, da für jeden neuen Schlüsselaustausch die Netztopologie durch die EPs neu erkundet wird. Deshalb beruht die Wegerkundung von MuCKE auf dieser Idee und löst dabei das Problem mit der schlechten Skalierbarkeit.
Vorweg noch ein wichtiges Detail: Jeder Schlüsselaustausch ist mit einer ID gekennzeichnet. Alle Pakete, die am Schlüsselaustausch beteiligt sind, enthalten sie und sind somit eindeutig zu einem Schlüsselaustausch zuzuordnen. Die ID wird dabei vom Initiator zufällig bestimmt.

4.1.1 Erkundungspakete

Die Aufgabe der EPs ist es, mehrere Wege zu finden. Damit dieser Ansatz gut skaliert, wird das Fluten der Pakete modifiziert. Die Idee von MuCKE ist die EPs begrenzt lokal zu fluten, was mithilfe eines Kontingents gelöst werden soll. Die EPs werden also solange geflutet, bis ein Knoten ein EP mit Kontingent null erhält. Dann wird das EP weiter zu seinem Empfänger geroutet. Das Kontingent sagt also aus, wie oft ein EP für das Fluten weitergeleitet werden darf. Dadurch kann die Netzlast, die das Protokoll auslöst, reguliert werden.
Des Weiteren wird jeder Knoten, der ein EP erhält und verarbeitet, diesem EP Informationen anhängen. Die Informationen bestehen dabei aus zwei Teilen. Der erste Teil enthält dabei die Adresse des Knotens. Im zweiten Teil befindet sich die Sicherheitswahrscheinlichkeit der Kante, über welche der Knoten das EP empfangen hat. Sollten bei dem Fluten oder Routen der EPs Kreise auftreten, dann kann dies bei der Verarbeitung erkannt werden. Dazu überprüft jeder Knoten, ob er bereits in den gesammelten Informationen vorkommt. Ist dies der Fall, dann existiert ein Kreis. Anstatt nun das Paket aufgrund des Kreises zu verwerfen, verfolgt MuCKE einen anderen Ansatz. Durch das Verwerfen solch eines EPs gehen bereits gesammelte Informationen verloren, was zu weniger Pfaden führen kann. Deswegen wird der Kreis aus den Informationen entfernt und das EP weiter verarbeitet. Das Verfahren der EP-Verarbeitung ist im Algorithmus 1 beschrieben.

4.1.2 Antwort-Erkundungspakete

Empfängt ein Knoten nun ein EP, was an ihn adressiert war, dann wird er darauf mit einem Antwort-Erkundungspaket (AEP) antworten. Ein AEP enthält in erster Linie alle Informationen, die mit dem dazugehörigen EP gesammelt worden sind. Jedes AEP stellt somit einen gefunden Pfad dar. Außerdem werden die AEPs durchnummeriert, um die gefundenen Pfade eindeutig identifizieren zu können. Verzichtet man auf die Nummerierung, dann können sich beide Teilnehmer nicht darauf verständigen, welche Pfade für den Schlüsselaustausch genutzt werden sollen.
Damit bereits nach einem Nachrichtenaustausch sowohl die Pfade entdeckt als auch die Zufallszahlen über ihren jeweiligen Pfad versendet worden sind, werden die Zufallszahlen den AEPs hinzugefügt. Dadurch wird jeweils ein Nachrichtenaustausch pro Pfad gespart, was sowohl für die Skalierbarkeit als auch für die Laufzeit relevant ist. Der Nachteil ist dabei, dass nur Bob bei dem Erzeugen der Zufallszahlen beteiligt ist. Alice bringt keine Information mit ein.
Die AEPs werden dabei über denselben Pfad zurück zu Alice gesendet, der mithilfe des dazugehörigen EPs gefunden wurde. Die Pfadinformation ist bereits in den AEPs enthalten und muss dementsprechend nicht zusätzlich angefügt werden. Des Weiteren ist dadurch jeder Knoten des Pfades in der Lage seine Informationen, die er in das

Algorithmus 1 EP-Verarbeitung

```
function EPPROC(EP P, Kontingent k, Sender a, Last-Hop w, Knoten v)
    if Knoten v kommt in P bereits vor then                ▷ Es existiert eine Schleife
        Entferne den Suffix der Informationen nach v
    else
        Füge die Adresse von v zu P hinzu
        Füge die Sicherheitswahrscheinlichkeit der Kante (w, v) zu P hinzu
    end if
    if k = 0 then
        Route P zu Bob
    else
        R ← N(v) \ {w, a}
        i ← 1
        while k > 0 do
            Wähle zufällig u ∈ R
            R ← R \ u
            if R = ∅ then
                k_i ← k − 1
            else
                Wähle zufällig k_i ∈ {0, ..., k − 1}
            end if
            k ← k − k_i − 1                    ▷ Jedes geflutete EP kostet 1 Kontingent
            P_i ← P
            Setze das Kontingent von P_i auf k_i
            Sende P_i an u.
            i ← i + 1
        end while
    end if
end function
```

EP geschrieben hat, zu überprüfen. Stimmen die Informationen nicht überein, wird das AEP aufgrund eines Manipulationsverdachts verworfen. Zum anderen fließen die Pfadinformationen mit in die Schlüsselaushandlung ein. Ist also ein Angreifer in der Lage den Inhalt eines AEPs zu verändern, ohne dass dies die Knoten auf dem Pfad erkennen (z.B. indem er die Informationen der Knoten ändert, die bereits das AEP gesehen haben), dann haben Alice und Bob verschiedene Informationen. Dies führt zu einem unterschiedlichen gemeinsamen Geheimnis, was den Abbruch des Schlüsselaustauschs zur Folge hat. Dadurch kann also zu einem gewissen Grad die Datenintegrität gesichert werden. Würde man die AEPs einfach durch das Netz routen, würden einerseits Knoten, die eventuell gar nicht an dem Schlüsselaustausch beteiligt sind, an Pfadinformationen und somit an die Zufallszahlen gelangen. Andererseits könnte die eben beschriebene Überprüfung der Daten nicht durchgeführt werden. Beides wäre also ein Verlust der Sicherheit, weswegen dies keine Option ist.

4.1.3 Abschluss-Pakete

Das Abschluss-Paket (AP) ist ein Paket, welches den Multipath-Schlüsselaustausch beendet. Ein weiterer Nachrichtenaustausch zwischen den beiden Teilnehmern wird benötigt. Sie müssen sich verständigen, welche Pfade nun für den Schlüsselaustausch genutzt werden. Bob hat eventuell mehr Pfade als Alice gefunden, weil versendete AEPs auf dem Weg verloren gegangen sind. Alice entscheidet dabei, welche Pfade genutzt werden. Sie wird dabei alle gefundenen Pfade nutzen, was die Sicherheit nur verbessern kann. Nimmt sie einen Pfad hinzu, der kompromittiert ist, dann sind die Pfadinformationen dieses Pfades dem Angreifer bekannt. Er müsste aber noch alle weiteren Pfade kompromittieren, um an alle Informationen zu gelangen, die für die Schlüsselableitung genutzt werden. Das Hinzufügen kompromittierter Pfade verschlechtert also die Sicherheit nicht. Sollte ein Angreifer den Inhalt des APs auf dem Weg manipulieren, dann führt dies (wie beim AEP) zu einem unterschiedlichen gemeinsamen Geheimnis, was ebenfalls einen Abbruch des Schlüsselaustauschs zur Folge hat.

4.1.4 Acknowledge-Paket

Das Acknowledge-Paket (ACK) ist Bobs Antwort auf ein AP. Dabei enthält das ACK bereits eine Information, die mithilfe des neu abgeleiteten Schlüssels berechnet wurde. Alice kann mithilfe dieser Information dann verifizieren, ob Bobs Schlüssel mit ihrem Schlüssel übereinstimmt. Ist dies nicht der Fall, wird die Schlüsselaushandlung beendet. Durch das Anhängen dieser Information kann das Verifizieren des Schlüssel vollzogen werden, ohne einen erneuten Nachrichtenaustausch zu initiieren.

4.1.5 Nachrichtenaustausch

Möchte Alice nun einen Schlüssel mit Bob aushandeln, dann wird sie die Rolle des Initiators übernehmen und im ersten Schritt EPs mit Bob als Empfänger an ihre direkten Nachbarn senden. Diese EPs werden dann solange geflutet (und dadurch dupliziert), bis sie ein Kontingent von null haben. Dann werden sie weiter zu Bob geroutet. Sobald Bob eines dieser EPs erhält, erstellt er ein passendes AEP und schickt dies (auf derselben Route wie das dazugehörige EP) zurück zu Alice. Diesen Mechanismus führt er für alle EPs aus, die er erhält. Nachdem Alice den Schlüsselaustausch initiiert hat und somit die EPs versendet hat, wird sie eine gewisse Zeit t warten. Innerhalb dieses Zeitraumes wird sie alle AEPs, die zu dem Schlüsselaustausch gehören, entgegennehmen. Ist die Zeit

abgelaufen, wird sie keine weiteren AEPs entgegennehmen. Sie erstellt ein AP und fügt dort alle IDs der AEPs, die sie erhalten hat, hinzu. Dieses AP sendet sie nun an Bob. Erhält Bob solch ein AP, dann wird er darauf mit einem ACK antworten. Außerdem weiß Bob nun, dass Alice keine weiteren AEPs entgegennehmen wird. Deswegen wird er selber nicht mehr auf eingehende EPs, die zu diesem Schlüsselaustausch gehören, antworten. Damit ist die Erkundungsphase beendet.
Um die gefundenen Pfade und die dadurch gesammelten Informationen nicht zu verwerfen, sollte ein AP nicht bei Bob ankommen, wird Alice versuchen das AP zuzustellen. Mithilfe des ACKs erkennt sie, ob Bob ihr AP erhalten hat. Sollte sie innerhalb einer gewissen Zeit kein ACK erhalten, wird sie das AP erneut versendet. Nach mehreren fehlgeschlagenen Versuchen wird dieser Mechanismus und somit die Schlüsselaushandlung abgebrochen. Sollte hingegen das Zustellen des APs erfolgreich sein, kann sie mithilfe der im ACK enthaltenen Information den Schlüssel verifizieren.
Um diesen Multipath-Nachrichtenaustausch nochmals zu verdeutlichen betrachte man Abbildung 4.1.

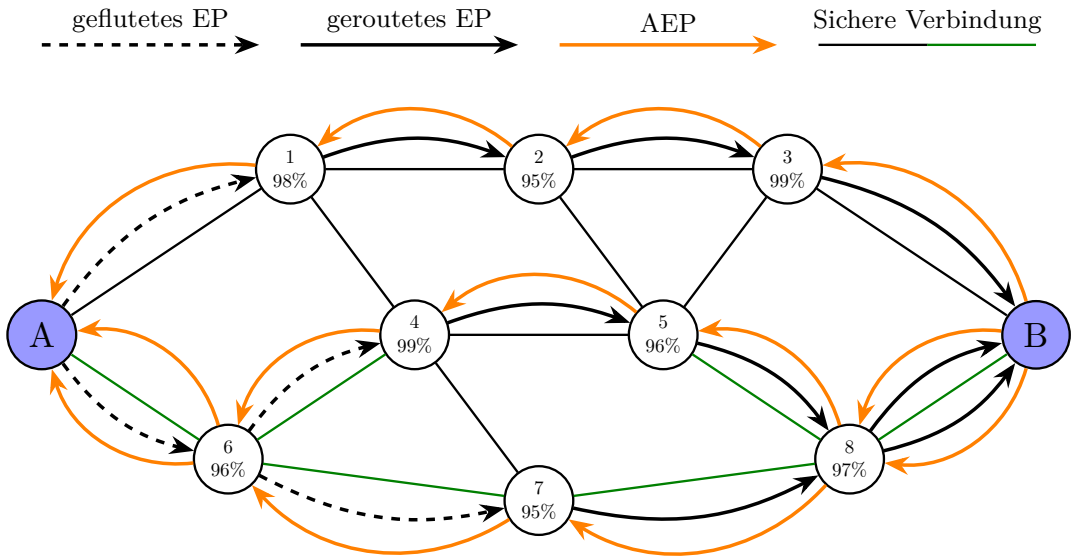


Abbildung 4.1: Eine Schlüsselaushandlung zwischen Alice und Bob über drei Pfade. Schwarze Verbindungen haben eine Sicherheit von 90 %, grüne von 99 %.

Zuerst wurden von Alice zwei EPs geflutet. Das EP, was an Knoten 1 gesendet wurde, hatte ein Kontingent von null. Dementsprechend hat dieser Knoten es nun weiter in Richtung Bob geroutet. So wurde also ein erster Pfad gefunden. Das zweite EP, was von Alice ausging, wurde an den Knoten 6 mit einem Kontingent von zwei versendet. Der Knoten 6 hat nun (durch Zufall) jeweils ein EP mit Kontingent null an Knoten 4 und 7 gesendet. Diese haben dann jeweils die EPs zu Bob geroutet, wodurch der zweite und dritte Pfad entdeckt worden sind. Bob hat also drei EPs erhalten, auf welche er jeweils mit einem dazugehörigen AEP antwortet. Diese AEPs gehen nun ihren Pfad entlang zurück zu Alice. Zu beachten ist, dass durch das von Alice an den Knoten 6 gesendete EP mehrere Pfade gefunden worden sind. Hingegen EPs mit einem Kontingent von null, bei denen man weiß, dass hierdurch maximal ein Pfad entdeckt werden kann, kann man bei geflutete EPs nicht wissen, wie viele Pfade dadurch noch entdeckt werden können. Abbildung 4.2 zeigt ein Sequenzdiagramm von einem möglichen Nachrichtenaustausch zu dem in Abbildung 4.1 gezeigten Szenario.

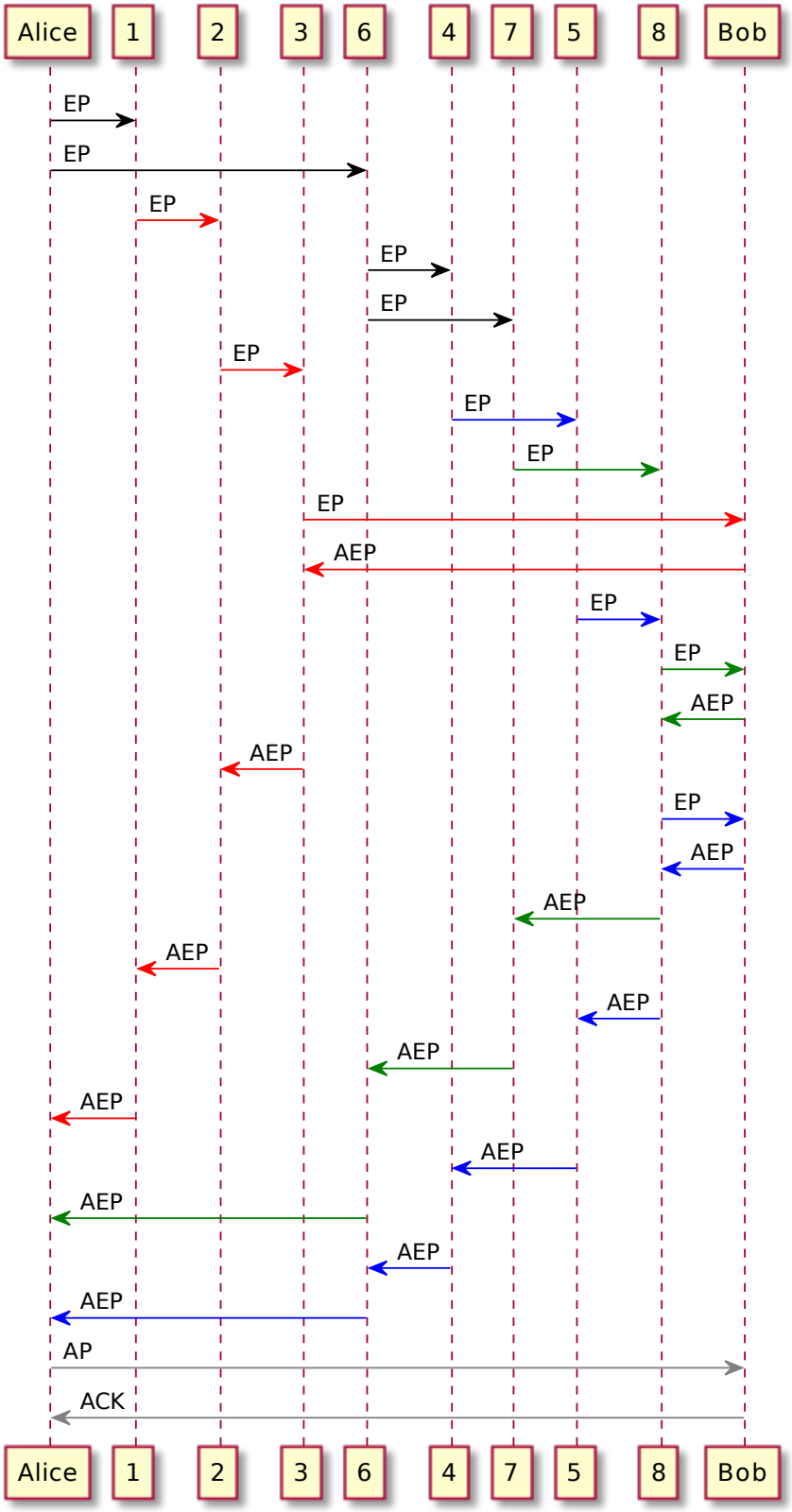


Abbildung 4.2: Der Multipath-Nachrichtenaustausch für die Beispielkommunikation in der Abbildung 4.1. Schwarze EPs stehen für geflutete EPs. Die gefärbten Pakete stellen die drei entdeckten Pfade dar.

4.2 Ableiten des Schlüssels

In diesem Unterabschnitt wird nun die Schlüsselableitung beschrieben. Zuerst wird dabei die Schlüsselableitung beschrieben, bei der noch kein gemeinsames Geheimnis zwischen Alice und Bob existiert. Darauf aufbauend wird das Erneuern des Schlüssels beschrieben.

4.2.1 Die erste Schlüsselableitung

Hier wird nun die Schlüsselableitung betrachtet, bei der zwischen Alice und Bob noch kein gemeinsames Geheimnis existiert. Es wird angenommen, dass durch die Erkundungsphase n Pfade $W_1, \ldots, W_n$ entdeckt worden sind. Zu jedem Pfad kennen Alice und Bob eine Zufallszahl r_i und die dazugehörige Information I_i. Die Information eines Pfades wird dabei für die Schlüsselableitung folgendermaßen aus den gewonnen Informationen abgeleitet:

$$I_i := (a, v_1) \parallel p_{(a,v_1)} \parallel v_1 \parallel p_{v_1} \parallel \cdots \parallel (v_k, b) \parallel p_{(v_k,b)} \tag{4.1}$$

für $W_i = (V_i, E_i)$ mit $\{a, v_1, \ldots, v_k, b\} = V_i$ und $\{(a, v_1), \ldots, (v_k, b)\} = E_i$. Der Knoten a ist dabei Alice, b Bob.
Dieses gemeinsame Wissen der Zufallszahlen und Informationen der Pfade wird nun genutzt um mithilfe einer Key Derivation Function (KDF) einen ersten Multipath-Schlüssel K_{MKR} abzuleiten:

$$K_{MKR} := KDF(r_1 \parallel \cdots \parallel r_n \parallel I_1 \parallel \cdots \parallel I_n). \tag{4.2}$$

Diese Art der Schlüsselableitung kann nur funktionieren, wenn Alice und Bob die Reihenfolge der Pfade festgelegt haben. Dies ist aber gegeben: Bob nummeriert die AEPs aufsteigend durch. Durch das von Alice an Bob gesendete AP wissen danach beide, welche Pfade sie nutzen und haben durch die aufsteigende Nummerierung der AEPs bereits eine Reihenfolge gegeben. Anstelle der Konkatenation könnten auch kommutative Bitoperationen (z.B. die XOR-Verknüpfung) genutzt werden. Durch kommutative Bitoperationen sinkt aber die Entropie. Die Größe der Zufallszahlen befindet sich in der Regel im Bereich von 32 B bis 64 B. Durch die Verknüpfung hätte die Bitfolge, die am Ende in die KDF fließt, dementsprechend eine maximale Entropie von 64 B. Bei der Konkatenation wird die Entropie durch die Anzahl der Pfade dominiert. Für jeden weiteren gefundenen Pfad steigt die Entropie der Bitfolge, die in die KDF fließt. Der gemeinsame Schlüssel bietet dementsprechend eine höhere Sicherheit.
Nach dem Multipath-Schlüsselaustausch wird zusätzlich ein IKEv2-Schlüsselaustausch ausgeführt. Dadurch kann ein zweiter Schlüssel K_{IKE} abgeleitet werden. Diese beiden Schlüssel werden nun folgendermaßen zum Master-Schlüssel $K_{A,B}$ vereint:

$$K_{A,B} := KDF(K_{MKR} \parallel K_{IKE}). \tag{4.3}$$

Eine andere Alternative den Multipath-Schlüssel mit dem IKEv2-Protokoll zu verbinden bietet der RFC 8784. Dieser wurde bereits im Unterabschnitt 3.6.3 beschrieben. Dieser Standard bietet die Möglichkeit einen vorher ausgehandelten Schlüssel mit in die IKEv2-Schlüsselaushandlungen einzubeziehen. Da aber auch hier der Multipath-Schlüssel erst nach der Authentisierungsphase des IKEv2-Protokolls miteinbezogen wird, bietet dies keinen Sicherheitsvorteil zu der in der Gleichung 4.3 gezeigten Option.
Am Ende haben Alice und Bob ein gemeinsames Geheimnis abgeleitet, welches die Eigenschaften des IKEv2-Schlüsselaustausch mit der Quantenresistenz des Multipath-Schlüsselaustauschs vereint.

4.2.2 Schlüssel-Update

Möchte man nun zu einem späteren Zeitpunkt wieder einen Schlüssel zwischen Alice und Bob ableiten, so wird das Protokoll erneut ausgeführt. Dadurch werden aufgrund der Dynamik des Netzes und des eingebrachten Zufalls bei der EP-Verarbeitung (siehe Algorithmus 1) mit hoher Wahrscheinlichkeit andere Knoten bzw. Kanten für die Übertragung genutzt. Die dadurch gewonnen Informationen fließen dann in den neu abgeleiteten Schlüssel K_{neu} mit ein. Anstatt jetzt nur den neuen Schlüssel K_{neu} zu nutzen (und somit den alten Schlüssel $K_{A,B}$ zu verwerfen), wird MuCKE das gemeinsame Geheimnis mithilfe des neuen Schlüssels erneuern:

$$K_{A,B} := KDF(K_{A,B} \parallel K_{neu}). \tag{4.4}$$

Dadurch sind die in der Vergangenheit gesammelten Informationen, die in die damaligen Schlüsselaushandlung eingeflossen sind, auch in den jetzige Schlüsselaushandlung genutzt worden. Die Sicherheit des gemeinsamen Geheimnisses steigert sich also.
Diese Art des Schlüssel-Updates birgt aber auch Risiken für die Robustheit. Sollte es bei dem Erneuern des Schlüsselaustauschs zu dem Fall kommen, dass einer der beiden Teilnehmer den Schlüssel erneuert, der andere nicht, dann ist dieser Mechanismus nicht mehr synchronisiert. Nimmt man nun o.B.d.A. an, dass Alice den Schlüssel erneuert hat, Bob nicht, dann führt selbst ein erneuter Schlüsselaustausch nicht zu einem gemeinsamen Geheimnis. Das aktuelle gemeinsame Geheimnis $K_{A,B}$ ist verschieden, wodurch die Ableitung des neuen gemeinsamen Geheimnisses auch verschieden sein wird. Man benötigt dementsprechend einen Mechanismus, der das gemeinsame Geheimnis wiederherstellt, ohne dabei die in der Vergangenheit gesammelten Informationen komplett zu verwerfen.
Im Folgenden wird der aktuelle Schlüssel als K_n bezeichnet, wobei n den aktuellen Status des Schlüssel-Updates darstellt. Der Begriff Fehler im Zusammenhang mit dem Schlüssel-Update bedeutet, dass einer der beiden Teilnehmer den Schlüssel erneuert hat, der andere nicht. Angenommen das Erneuern des Schlüssels ist nicht mehr synchronisiert, dann gibt es nur zwei Möglichkeiten, die solch ein Ereignis hervorrufen können:

1. Einer der beiden Teilnehmer hat ein Schlüssel-Update initiiert. Dabei ist ein Fehler aufgetreten und o.B.d.A. hat Alice den Schlüssel erneuert, Bob nicht. Alice nutzt nun den Schlüssel K_{n+1}, Bob aber K_n.

2. Beide Teilnehmer haben gleichzeitig ein Schlüssel-Update initiiert.

   a) Das eine Schlüssel-Update ist erfolgreich. Bei dem anderen Schlüssel-Update tritt ein Fehler auf und o.B.d.A. hat Alice den Schlüssel zweimal erneuert, Bob aber nur einmal. Alice nutzt nun den Schlüssel K_{n+2}, Bob aber K_{n+1}.

   b) Das eine Schlüssel-Update wurde abgebrochen. Bei dem anderen Schlüssel-Update tritt ein Fehler auf und o.B.d.A. hat Alice den Schlüssel einmal erneuert, Bob nicht. Alice nutzt nun den Schlüssel K_{n+1}, Bob aber K_n.

   c) Bei beiden Schlüssel-Updates kommt es zu einem Fehler.

      i. Dabei hat o.B.d.A. Alice zwei Mal den Schlüssel erneuert, Bob nicht. Alice nutzt nun den Schlüssel K_{n+2}, Bob aber K_n.

      ii. Beide haben jeweils einen neuen Schlüssel abgeleitet, wobei diese verschieden sind. Alice besitzt also einen Schlüssel K^A_{n+1}, Bob K^B_{n+1} mit $K^A_{n+1} \neq K^B_{n+1}$.

Der Fall, dass mehrere Schlüssel-Updates hintereinander fehlschlagen, kann nicht eintreten. Denn sobald eine Erneuerung des Master-Schlüssel $K_{A,B}$ abgeschlossen wurde, wird im Anschluss mithilfe eines Nachrichtenaustauschs überprüft, ob der erneuerte Schlüssel übereinstimmt. Sollte bei diesem Schlüssel-Update bereits ein Fehler aufgetreten sein, dann wird dies dadurch erkannt und dementsprechend reagieren.
Ist nun solch ein Fehler aufgetreten und erkannt worden, dann muss dieser Fehler rückgängig gemacht werden. Man benötigt also einen Wiederherstellungs-Mechanismus. Das Erneuern des Schlüssels kann maximal um zwei Stufen verschoben sein. Alice und Bob speichern sich also neben dem aktuellen Schlüssel auch die letzten zwei genutzten Schlüssel. Empfängt nun o.B.d.A. Alice eine Nachricht aus diesem Nachrichtenaustausch, der den Schlüssel überprüft, und sie kann diese nicht mit dem aktuellen Schlüssel entschlüsseln, dann wird sie zuerst versuchen diese Nachricht mit den letzten beiden Schlüsseln zu entschlüsseln. Gelingt dies für einen dieser beiden Schlüssel, dann kann ein Fallback auf diesen Schlüssel ausgeführt werden und die Schlüsselaushandlung ist wieder synchronisiert. Andernfalls weiß man, dass Bob einen Fehler gemacht hat. Alice sendet nun ein Recovery-Paket (RP) an Bob. Ein RP ist ein spezielles Paket, was (neben den Paketen für die Überprüfung des Schlüssels) den Wiederherstellungs-Mechanismus auslöst. Es enthält eine Information, die aus den letzten funktionsfähigen Schlüssel abgeleitet wurde. Bob selber kann das RP nicht mit seinem aktuellen Schlüssel entschlüsseln. Aber einer der beiden letzten Schlüssel wird funktionieren und die enthaltene Information wird übereinstimmen. Auf diesen Schlüssel wird dann ein Fallback ausgeführt und der Schlüsselaustausch ist wieder synchronisiert. Dieser Ablauf ist formal im Algorithmus 2 beschrieben.

Algorithmus 2 Wiederherstellungs-Mechanismus

function KeyRecovery(Sender b, Nachricht M, Schlüssel (K_n, K_{n-1}, K_{n-2}))
  if `decrypt`(M, K_n) funktioniert **then**
    return ▷ Schlüssel sind synchronisiert
  else if `decrypt`(M, K_{n-1}) funktioniert **then**
    Setzte den aktuelle Schlüssel auf K_{n-1}
    Verwerfe K_n
  else if `decrypt`(M, K_{n-2}) funktioniert **then**
    Setzte den aktuelle Schlüssel auf K_{n-2}
    Verwerfe K_n und K_{n-1}
  else ▷ Der andere Teilnehmer hat einen Fehler gemacht
    Sende RP P verschlüsselt mit K_n an b
  end if
end function

Sollte es dazu kommen, dass trotz des Wiederherstellungs-Mechanismus der Schlüssel nicht wiederhergestellt werden kann, dann liegt diesem Problem ein anderer Fehler (z.B. ein Speicherfehler) zu Grunde. Dann muss, falls beide Instanzen wieder einen Schlüssel ableiten wollen, ein komplett neuer Schlüssel manuell ausgehandelt werden.

Sicherheit dieses Mechanismus

In diesem Abschnitt soll diskutiert werden, ob und wie dieser Mechanismus genutzt werden kann, um MuCKE zu stören. Manipuliert ein Angreifer ein Paket, welches zum Verifizieren des Schlüssels genutzt wird, dann wird der Mechanismus ausgelöst. Alice kann dann dieses Paket nicht erfolgreich entschlüsseln, weswegen sie von einem Fehler

auf Bobs Seite ausgeht. Sie sendet daraufhin ein RP an Bob. Bob kann dieses RP aber mit dem ersten (und somit aktuellen) Schlüssel entschlüsseln. Da dies im Normalfall nicht passieren sollte, erkennt er diesen Fall als einen potenziellen Angriff.
Sollte der Angreifer den aktuellen Schlüssel kompromittiert haben, dann kann er dadurch die Information, die in dem RP enthalten ist, ableiten. Er ist damit also in der Lage valide RPs zu erstellen. Wird jetzt ein Schlüssel-Update initiiert, dann gibt es dafür drei Möglichkeiten:

- Das Schlüssel-Update ist erfolgreich. Dann wird der aktuelle Schlüssel und somit die Information des RPs erneuert. Wenn er jetzt das RP an Alice oder Bob sendet, dann kann dieses zwar entschlüsselt werden, aber die Information stimmt nicht überein, was als potenzieller Angriff gewertet wird. Das RP bietet dem Angreifer also keinen Mehrwert.

- Der Schlüssel ist nach dem Update verschieden. Dann kann er mithilfe seines RPs den Mechanismus auslösen und den zuletzt genutzten Schlüssel wiederherstellen. Dies wäre aber durch das Protokoll selbst initialisiert worden und bietet dem Angreifer deshalb keinen Mehrwert.

- Das Schlüssel-Update ist fehlgeschlagen. Dann ist er in derselben Situation wie eingangs erwähnt. Die aktuelle Kommunikation kann er weiter abhören.

Das Stören des Schlüsselaustauschs ist also für einen Angreifer die einzige Möglichkeit weiter die Verbindung abzuhören, sollte er das aktuelle gemeinsame Geheimnis kennen. Wie das Protokoll gestört werden kann, wird im Unterabschnitt 5.1.2 genauer diskutiert. Sollte die erneute Schlüsselaushandlung zu oft fehlschlagen, dann kann dies auch als potenzieller Angriff gewertet werden.

4.3 Automatisierte Bewertung der erreichten Sicherheit

Eine weitere Funktionalität des Protokolls ist es, die Sicherheit des ausgehandelten Schlüssels zu berechnen. Dazu wird speziell der Schlüssel K_{MKR} betrachtet, da diese Sicherheit von der Netztopologie und der gewählten Wege abhängig ist. Die Sicherheit des Schlüssels K_{IKE} wird als konstant angenommen, da für die Sicherheit die Existenz von kryptographisch relevanten Quantencomputern maßgeblich ist. Im Folgenden wird K für den Schlüssel K_{MKR} geschrieben. Außerdem ist mit der Sicherheit eines Knoten bzw. einer Kante die Sicherheitswahrscheinlichkeit gemeint.
Zunächst muss geklärt werden, welche Informationen zu diesem Zeitpunkt vorhanden sind. Durch die EPs wurden n Wege ermittelt und für die Schlüsselaushandlung genutzt. Die Menge aller Wege wird als $\mathfrak{W}$ bezeichnet. Zu jedem dieser Wege ist jeder Knoten und jede Kante bekannt. Außerdem wurde direkt die Sicherheit der Kanten mitgesendet. Um die Sicherheit der Knoten zu bestimmen, müssen Zertifikate genutzt werden. Dafür benötigt man quantenresistente Signaturverfahren. Die Klasse der hashbasierten Kryptographie bietet solche Verfahren. Damit können solche Zertifikate realisiert werden. Ein Standard, der dafür genutzt werden könnte, wird im Unterabschnitt 5.1.2 im Punkt 3 näher beschrieben.

4.3.1 Berechnung der Schwachstellen

Ausgehend von diesen Informationen müssen die Schwachstellen $\mathfrak{S}$ berechnet werden. Zur Erinnerung: Eine Schwachstelle $S \in \mathfrak{S}$ ist eine Menge von Knoten und Kanten mit

der Eigenschaft, dass es mindestens zwei Wege gibt, die alle Elemente von S gemeinsam haben. Für die Berechnung der Schwachstellen werden Äquivalenzklassen gebildet. Dazu betrachtet man im ersten Schritt alle Pfade $\mathfrak{W}$ und merkt sich zu jedem Knoten bzw. jeder Kante, an welchen Pfaden der Knoten bzw. die Kante beteiligt sind. Man erhält also eine Liste von Knoten und Kanten, wobei jedem Eintrag die beteiligten Pfade zugeteilt sind. Über diese Liste bildet man nun folgende Äquivalenzrelation:

Zwei Elemente sind genau dann äquivalent, wenn sie denselben Pfaden angehören.

Aufgrund dieser Äquivalenzrelation können nun Äquivalenzklassen gebildet werden. Alle Äquivalenzklassen von Knoten bzw. Kanten, die zu mindestens zwei Pfaden zugeordnet wurden, bilden dann die Schwachstellen der Menge $\mathfrak{W}$ und werden in der Menge $\mathfrak{S}$ gespeichert. Zu bemerken ist, dass alle Schwachstellen aus $\mathfrak{S}$ aufgrund der Eigenschaften von Äquivalenzklassen paarweise disjunkt sind.
Um die Berechnung der Schwachstellen zu verdeutlichen betrachte man Abbildung 4.1. In diesem Beispiel gibt es drei Pfade $\mathfrak{W} = \{W_1, W_2, W_3\}$. Dabei werden zuerst die Knoten a und b (Alice und Bob) aus den Pfaden entfernt, da diese Knoten für die Berechnung uninteressant sind. Man kann annehmen, dass Alice und Bob beide sicher sind, da sie untereinander einen Schlüssel austauschen wollen und sich somit gegenseitig vertrauen.

$$
\begin{aligned}
W_1 &= (\{v_1, v_2, v_3\}, \{(a, v_1), (v_1, v_2), (v_2, v_3), (v_3, b)\}) \\
W_2 &= (\{v_4, v_5, v_6, v_8\}, \{(a, v_6), (v_6, v_4), (v_4, v_5), (v_5, v_8), (v_8, b)\}) \\
W_3 &= (\{v_6, v_7, v_8\}, \{(a, v_6), (v_6, v_7), (v_7, v_8), (v_8, b)\}) .
\end{aligned}
$$

Betrachtet man nun die Knoten und Kanten des Netzes und teilt ihnen die dazugehörigen Pfade zu, dann entstehen folgende Tabellen:

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
W_1	W_1	W_1	W_2	W_2	W_2, W_3	W_3	W_2, W_3

Tabelle 4.1: die Knoten der Pfade aus $\mathfrak{W}$ mit den dazugehörigen Pfaden

(a, v_1)	(v_1, v_2)	(v_2, v_3)	(v_3, b)	(a, v_6)	(v_6, v_4)	(v_4, v_5)	(v_5, v_8)	(v_8, b)	(v_6, v_7)	(v_7, v_8)
W_1	W_1	W_1	W_1	W_2, W_3	W_2	W_2	W_2	W_2, W_3	W_3	W_3

Tabelle 4.2: die Kanten der Pfade aus $\mathfrak{W}$ mit den dazugehörigen Pfaden

Die dazugehörigen Äquivalenzklassen sind dann:

$$
\begin{aligned}
[W_1] := [v_1] &= (\{v_1, v_2, v_3\}, \{(a, v_1), (v_1, v_2), (v_2, v_3), (v_3, b)\}) \\
[W_2] := [v_4] &= (\{v_4, v_5\}, \{(v_6, v_4), (v_4, v_5), (v_5, v_8)\}) \\
[W_3] := [v_7] &= (\{v_7\}, \{(v_6, v_7), (v_7, v_8)\}) \\
[W_2, W_3] := [v_8] &= (\{v_6, v_8\}, \{(a, v_6), (v_8, b)\})
\end{aligned}
$$

Aus der Äquivalenzklasse $[W_2, W_3]$ resultiert dann eine Schwachstelle S_1 mit $S_1 = (\{v_6, v_8\}, \{(a, v_6), (v_8, b)\})$, wodurch die Menge der Schwachstellen $\mathfrak{S} = \{S_1\}$ gegeben ist.

4.3.2 Berechnung der Sicherheit

Mithilfe der Schwachstellen $\mathfrak{S}$ lässt sich nun die Sicherheit des Schlüssels K berechnen. Dazu werden zuerst folgende Ereignisse definiert:

$$\begin{aligned} A &:= \text{Schlüssel } K \text{ ist sicher,} \\ B_{\mathfrak{X}} &:= \text{Alle Schwachstellen aus } \mathfrak{X} \text{ sind kompromittiert, die restlichen nicht,} \end{aligned}$$

für ein $\mathfrak{X} \subseteq \mathfrak{S}$. Zu berechnen ist also die Wahrscheinlichkeit $P(A)$. Bevor man diese aber ableiten kann, muss zuerst die Wahrscheinlichkeit, dass eine Schwachstelle bzw. ein Pfad W sicher ist, berechnet werden. Diese lässt sich aus den bereits erworbenen Informationen folgendermaßen ableiten:

$$P(\text{Pfad bzw. Schwachstelle } W = (V, E) \text{ ist sicher}) := p_W = \prod_{v \in V} p_v \cdot \prod_{e \in E} p_e \tag{4.5}$$

Mithilfe dieser neu gewonnenen Information kann nun die Wahrscheinlichkeit $P(A)$ berechnet werden. Dazu unterscheidet man zwei Fälle. Im ersten Fall existieren keine Schwachstellen, d.h. $\mathfrak{S} = \emptyset$. Dann ergibt sich folgende Gleichung:

$$\begin{aligned} P(A) := p_K &= 1 - P(\overline{A}) \\ &= 1 - P(\text{Alle Pfade } W_1, \ldots, W_n \text{ sind kompromittiert}) \\ &= 1 - \left[\prod_{W \in \mathfrak{W}} (1 - p_W) \right] \end{aligned} \tag{4.6}$$

Ist die Menge der Schwachstellen nichtleer, dann ist die Betrachtung der Wahrscheinlichkeit $P(A)$ abhängig von den Schwachstellen. Der Grund dafür ist, dass durch das Kompromittieren einer Schwachstelle gleich mehrere Pfade kompromittiert werden. Dadurch müssen nun alle Möglichkeiten in Betracht gezogen werden, um $P(A)$ zu berechnen. Da die Elemente von $\mathfrak{S}$ paarweise disjunkt sind, gilt nach dem Satz der totalen Wahrscheinlichkeit:

$$P(A) = p_K = 1 - P(\overline{A}) = 1 - \left[\sum_{\mathfrak{X} \in \mathfrak{P}(\mathfrak{S})} P(\overline{A} | B_{\mathfrak{X}}) \cdot P(B_{\mathfrak{X}}) \right] \tag{4.7}$$

mit

$$P(B_{\mathfrak{X}}) = \prod_{X \in \mathfrak{X}} (1 - p_X) \cdot \prod_{Y \in (\mathfrak{S} \setminus \mathfrak{X})} p_Y \tag{4.8}$$

$$P(\overline{A} | B_{\mathfrak{X}}) = \prod_{\substack{W \in \mathfrak{W} \\ \forall X \in \mathfrak{X}: W \cap X = (\emptyset, \emptyset)}} \left[1 - \left(p_W \cdot \prod_{\substack{Y \in (\mathfrak{S} \setminus \mathfrak{X}) \\ W \cap Y = Y}} \frac{1}{p_Y} \right) \right] \tag{4.9}$$

für ein $\mathfrak{X} \subseteq \mathfrak{S}$. Mithilfe dieser Gleichungen lässt sich nun die Wahrscheinlichkeit $P(A)$, welche maßgeblich von den Schwachstellen $\mathfrak{S}$ abhängig ist, berechnen. Man betrachte dabei erneut Abbildung 4.1. In diesem Beispiel wurden drei Pfade $\mathfrak{W} = \{W_1, W_2, W_3\}$ entdeckt und für die Schlüsselaushandlung genutzt. Die benötigten Sicherheitswahrscheinlichkeiten sind bekannt. Zuerst müssen also die Schwachstellen berechnet werden. Dies wurde bereits im Unterabschnitt 4.3.1 für dieses Beispiel gezeigt. Es existiert eine Schwachstelle $S_1 = (\{v_6, v_8\}, \{(a, v_6), (v_8, b)\})$. Als nächsten werden die Wahrscheinlichkeiten der einzelnen Pfade und Schwachstellen berechnet:

- $p_{W_1} = p_{(a,v_1)} \cdot p_{v_1} \cdot p_{(v_1,v_2)} \cdot p_{v_2} \cdot p_{(v_2,v_3)} \cdot p_{v_3} \cdot p_{(v_3,b)} \approx 60\,\%$
- $p_{W_2} = p_{(a,v_6)} \cdot p_{v_6} \cdot p_{(v_6,v_4)} \cdot p_{v_4} \cdot p_{(v_4,v_5)} \cdot p_{v_5} \cdot p_{(v_5,v_8)} \cdot p_{v_8} \cdot p_{(v_8,b)} \approx 77\,\%$
- $p_{W_3} = p_{(a,v_6)} \cdot p_{v_6} \cdot p_{(v_6,v_7)} \cdot p_{v_7} \cdot p_{(v_7,v_8)} \cdot p_{v_8} \cdot p_{(v_8,b)} \approx 85\,\%$
- $p_{S_1} = p_{(a,v_6)} \cdot p_{v_6} \cdot p_{v_8} \cdot p_{(v_8,b)} \approx 91\,\%$

Da $\mathfrak{S} = \{S_1\} \neq \emptyset$ gilt, muss nun über die Potenzmenge von $\mathfrak{S}$ summiert werden:

$$\begin{aligned} P(A) &= 1 - \left[P(\overline{A}|B_\emptyset) + P(\overline{A}|B_{\{S_1\}})\right] \\ &= 1 - \left[(1 - p_{W_1}) \cdot \left(1 - \frac{p_{W_2}}{p_{S_1}}\right) \cdot \left(1 - \frac{p_{W_3}}{p_{S_1}}\right) \cdot p_{S_1} + (1 - p_{W_1}) \cdot (1 - p_{S_1})\right] \\ &\approx 96\,\%. \end{aligned}$$

An diesem Beispiel kann man also sehen, dass trotz vorhandener Schwachstellen das Nutzen mehrerer Pfade einen Sicherheitszuwachs bedeutet.

4.3.3 Limitationen dieser Bewertung

Die hier beschriebene automatisierte Bewertung der erreichten Sicherheit ist nicht allgemeingültig. Sie gibt unter bestimmten Rahmenbedingungen an, wie wahrscheinlich (oder unwahrscheinlich) es ist, dass der Multipath-Schlüssel K_{MKR} nicht kompromittiert ist. Die zugrundeliegenden Sicherheitswahrscheinlichkeiten und die Wahrscheinlichkeit $P(A)$ geben keine Auskunft darüber, ob der Knoten, die Kante oder der Schlüssel kompromittiert sind. Ein Angreifer könnte die Sicherheitswahrscheinlichkeiten seiner kompromittierten Elemente des Netzes künstlich erhöhen. Das Resultat wäre eine scheinbar erhöhte Sicherheit des Schlüssels, welche aber nicht die Realität widerspiegelt. Was außerdem die automatisierte Bewertung der erreichten Sicherheit nicht betrachtet, sind globale Schwachstellen. Für die Sicherheitsbewertung wird immer eine Momentaufnahme des Netzes betrachtet, indem nur ein Schlüsselaustausch stattgefunden hat. In der Realität ist dies nicht der Fall. Es könnten also Knoten bzw. Kanten in dem Netz existieren, über die häufig ein Schlüsselaustausch stattgefunden hat. Genau diese Knoten bzw. Kanten bilden dann eine globale Schwachstelle.
Auch für das Erneuern des Schlüssel liefert die automatisierte Bewertung keine korrekte Aussage. Denn der in Unterabschnitt 4.2.2 beschriebene Mechanismus zum Erneuern des Schlüssel nutzt sowohl den letzten Schlüssel als auch den neu abgeleiteten Schlüssel für die Schlüsselableitung. Um diese Sicherheit zu bewerten müsste man die genutzten Pfade in der Vergangenheit zeitlich betrachten, was wesentlich mehr Bedingungen und Abhängigkeiten mit sich führt. Aus diesem Grund kann die hierdurch bewertete Sicherheit nicht als Sicherheitswahrscheinlichkeit für die neu etablierte Verbindung zwischen Alice und Bob genutzt werden.
Ungeachtet dessen bietet die automatisierte Bewertung der erreichten Sicherheit unter der Annahme, dass die Sicherheitswahrscheinlichkeiten nicht manipuliert sind, eine zutreffende Aussage über die Sicherheit des neu abgeleiteten Schlüssels K_{MKR}.
Die automatisierte Bewertung der erreichten Sicherheit muss deswegen weiter untersucht werden, um die Abhängigkeit von globalen Schwachstellen und vorher ausgeführten Schlüsselaustauschen miteinzubeziehen.

4.4 Details der Implementierung

In diesem Abschnitt werden wichtige Details der Implementierung vorgestellt und deren Funktionalität begründet. Wie bereits im Abschnitt 2.4 angekündigt, wurde MuCKE in SOLID implementiert. Das Verfahren hat somit Zugriff auf Funktionalitäten von SOLID. Da MuCKE auch Pakete durch das Netz routen muss, wird im ersten Abschnitt erläutert, wie dies umgesetzt worden ist. Im Anschluss werden Parameter vorgestellt, welche die Funktionalität von MuCKE in großen Netzen sichern.

4.4.1 Routing von Paketen

Das Routing von Paketen ist ein wichtige Funktion, die mithilfe von SOLID realisiert wird. Die grundlegende Idee des Routings von Paketen in einem SOLID Netz ist im Abschnitt 2.4 beschrieben. Dabei müssen EPs mit einem Kontingent von null, APs und ACKs geroutet werden. Bei der Implementierung stellte sich aber heraus, dass es zu sehr großen Verzögerungen beim Versenden der ACKs kam. Die Problematik dabei entsteht aus dem Zusammenspiel von Netzaufbau und Schlüsselaushandlung. In dem getesteten Szenario wurde ein Netz ohne MuCKE aufgebaut. Danach wurde ein neuer Knoten 0 in das Netz hinzugefügt, welcher dann mithilfe von MuCKE die Verbindungen aufbaut. Wollte nun ein beliebiger Knoten ein Paket zu dem Knoten 0 routen, dann kam es zu folgendem Fehler: Das Paket wurde in Richtung des Knotens 0 geroutet. Auf diesem Weg entschied irgendwann ein Knoten, der eine Verbindung zum Knoten 1 hat, dass der Knoten 1 ein gutes Ziel zum Routen zum Knoten 0 ist. Man erinnere sich an die Ringstruktur des Netzes: Knoten 1 und 0 sollten in dieser Struktur eine Verbindung haben. Da jedoch Knoten 0 erst später hinzugefügt wurde, existierte diese Verbindung noch nicht. Knoten 1 routet dementsprechend das Paket in die entgegengesetzte Richtung, also zu Knoten 2. Knoten 2 sendet daraufhin wieder das Paket zum Knoten 1, da die Annahme der Verbindung weiterhin besteht. Das Paket pendelt solange zwischen Knoten 1 und 2, bis die Verbindung zwischen dem Knoten 1 und 0 aufgebaut ist. Dieser Fehler trat dann bei dem Versenden der ACKs auf. Beim Routen vom Knoten 0 zu einem beliebigen anderen Knoten traten keine Probleme auf. Um dieses Problem nun zu umgehen, werden sowohl die APs als auch die ACKs über einen bereits gefundenen Pfad geroutet. Dieser Pfad wird dabei zufällig aus den durch die Flutphase entdeckten Pfaden gewählt. Die Information des Pfades muss zusätzlich an diese Pakete angehängt werden, um die Pakete über diesem Pfad zu routen. Dies hat den Nachteil, dass sich die Paketgröße vergrößert. Es löst dafür das Problem des Routings und bietet noch eine weiteren Vorteil, der im Unterabschnitt 5.1.2 beschrieben wird.

4.4.2 Weitere Parameter

Um die Skalierbarkeit in Abhängigkeit der Größe des Netzes zu sichern wurden bei der Implementierung des Protokolls wichtige Parameter eingeführt. Diese Parameter werden in diesem Abschnitt vorgestellt und deren Funktionalität erläutert.

- `floodAllocation`: Dieser Parameter spiegelt das Startkontingent wider, welches beim initialen Fluten der EPs genutzt wird.
- `maxPeers`: Der Parameter regelt dabei die Anzahl der Nachbarn eines Knotens, die beim Fluten der EPs ausgewählt werden. Bezieht man dies auf den Algorithmus 1, dann wird die Größe der Menge R auf $|R| \leq$ `maxPeers` beschränkt.

- `timeWaitForFirstMsg`: Der Parameter regelt die Zeit, wie lange der Initiator auf das erste AEP wartet. Wird innerhalb dieser Zeit kein AEP empfangen, dann wird der Schlüsselaustausch abgebrochen.
- `timeFlood`: Dieser Parameter regelt die Zeit, wie lange nach dem ersten Empfangen des AEPs weitere AEPs entgegengenommen werden. Nach dieser Zeit wird die Phase des Flutens beendet.

Des Weiteren wurde die Größe der Zufallszahlen, die mit Hilfe der AEPs ausgetauscht werden, auf 32 B festgelegt.

Kapitel 5

Evaluierung

Die Evaluierung von MuCKE gliedert sich in zwei Teile. Zunächst wird die Umsetzung der in Kapitel 3 formulierten Anforderungen diskutiert. Einige ausgewählte Fragestellungen werden zusätzlich mithilfe einer Simulation näher untersucht und deren Resultate im zweiten Teil vorgestellt.

5.1 Betrachtung qualitativer Anforderungen

Im Folgenden wird die Umsetzung der verschiedenen Anforderungen aus Kapitel 3 betrachtet. Dies ist dabei in zwei Teile gegliedert: die Umsetzung der funktionalen Anforderungen und die Umsetzung der nichtfunktionalen Anforderungen.

5.1.1 Umsetzung funktionaler Anforderungen

In diesem Abschnitt werden die funktionalen Anforderungen, welche im Abschnitt 3.1 beschrieben sind, diskutiert.

1. **Schlüsselaustausch:** Die zentrale Funktionalität von MuCKE ist der Schlüsselaustausch zwischen zwei Netzknoten in einem zusammenhängenden WAN. Um diese Funktionalität zu diskutieren, betrachte man zuerst Abbildung 5.1. In dieser Abbildung ist der Zustandsautomat des Protokolls dargestellt. Dabei gibt es drei Möglichkeiten um in den Zustand „Abbruch“ zu gelangen:
   - Nach dem initialen Fluten wartet das Protokoll maximal `timeWaitForFirstMsg` Zeiteinheiten auf ein AEP. Ist in dieser Zeit kein AEP empfangen worden, dann wird der Schlüsselaustausch abgebrochen.
   - Sollte nach der Phase des Flutens das Zustellen des APs fehlschlagen, dann wird der Schlüsselaustausch ebenfalls abgebrochen.
   - Nachdem der Schlüssel abgleitet wurde, kann mithilfe der im ACK enthaltenen Information der Schlüssel überprüft werden. Sollte diese Überprüfung fehlschlagen, dann haben Alice und Bob verschiedene Informationen zur Schlüsselableitung genutzt, was auf manipulierte Pakete zurückzuführen ist. Der Schlüsselaustausch wird abgebrochen.

   All diese Fälle sind dabei entweder auf Probleme im Netz, wie etwa Paketverlust oder hohe Verzögerung, oder auf die Manipulation durch einen Angreifer zurückzuführen. In allen anderen Fällen findet ein Schlüsselaustausch statt. Dabei ist nicht garantiert, dass eine gewisse Anzahl an Pfaden genutzt wird. Sollte nur ein Pfad zum Schlüsselaustausch genutzt werden, dann bietet dieser Schlüsselaustausch nur einen minimalen Mehrwert. Der genutzte Pfad kann verschieden zu dem Pfad sein, über den die Pakete des IKEv2-Schlüsselaustauschs geroutet worden sind. Deswegen wird die Anzahl der Pfade, die für einen Schlüsselaustausch genutzt werden, im Abschnitt 5.2 weiter untersucht.

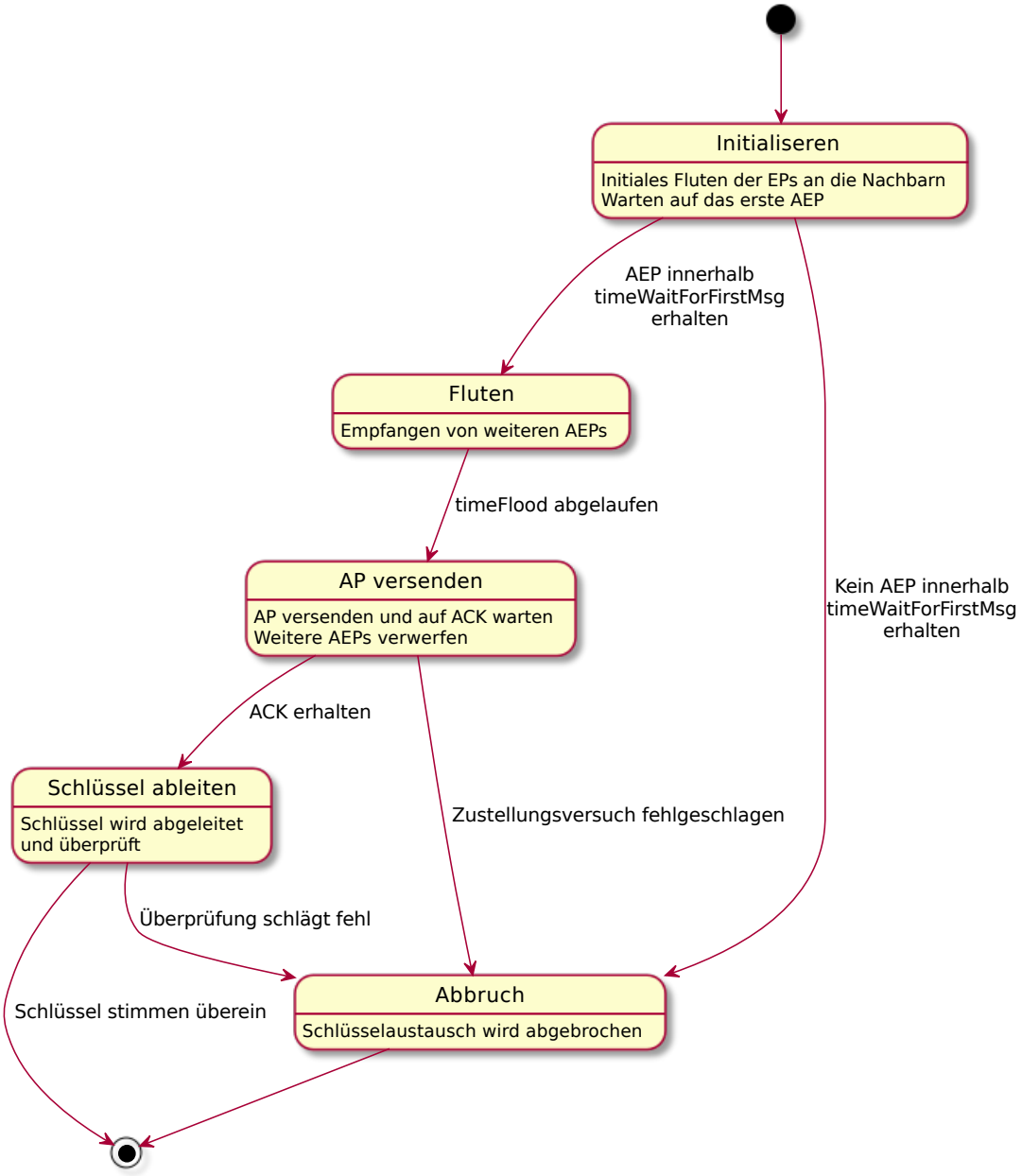


Abbildung 5.1: Das Zustandsdiagramm des MuCKE-Protokolls.

2. **Sicherheitsbewertung:** Durch das im Abschnitt 4.3 beschriebene Verfahren kann mithilfe der gesammelten Sicherheitswahrscheinlichkeiten der Kanten und die durch Zertifikate bestimmbaren Sicherheitswahrscheinlichkeiten der Knoten die Sicherheit des abgeleiteten Geheimnisses in Prozent berechnet werden. Für Zertifikate benötigt man asymmetrische Kryptographie. Da MuCKE aber auf asymmetrische Verfahren verzichten möchte, ist dies im ersten Moment nicht möglich. Im Unterabschnitt 2.2.1 wurden hashbasierte Signaturen vorgestellt. Ein möglicher Standard aus der Klasse der hashbasierten Signaturen, mit dem man quantenresistente Signaturen erzeugen kann, wird im folgenden Unterabschnitt 5.1.2 bei dem Punkt 3 diskutiert. Die Limitationen dieser Berechnung sind bereits im Unterabschnitt 4.3.3 diskutiert worden.

3. **Perfect Forward Secrecy:** Mithilfe des im Unterabschnitt 4.2.2 beschriebenen Mechanismus des Schlüssel-Updates kann das Protokoll den aktuellen Schlüssel erneuern. Im Unterunterabschnitt 4.2.2 wurde bereits gezeigt, dass ein erfolgreiches Erneuern des Schlüssels den alten Schlüssel für einen Angreifer nutzlos macht. Ein Angreifer muss also jeden neuen Schlüssel erneut kompromittieren um die neue Sitzung abzuhören. Die PFS ist dementsprechend erfüllt.

5.1.2 Umsetzung nichtfunktionaler Anforderungen

Im Folgenden werden nun die Anforderungen betrachtet, die nicht die Funktionalität, sondern die Qualität des Protokolls sichern sollen.

Skalierbarkeit

Um die Skalierbarkeit von MuCKE zu sichern, wurde bei der Entwicklung darauf geachtet, dass die Verarbeitung der Pakete höchstens linearen Rechenaufwand in Abhängigkeit von der Größe des Netzes bedeutet. Betrachtet man den Algorithmus 1, dann ist die Laufzeit linear abhängig von der Anzahl der Nachbarn eines Knotens. Da es dafür bei der Implementierung den Parameter `maxPeers` gibt, der die Anzahl der Nachbarn, die Pakete erhalten, begrenzt, ist die Laufzeit abhängig von diesem Parameter. Da `maxPeers` aber für ein gegebenes Netz festgelegt wird, ist dieser Parameter und somit die Laufzeit vom Algorithmus 1 konstant. Bei der Verarbeitung der AEPs muss der mitgesendete Pfad betrachtet werden, um den Empfängerknoten zu finden. Dies hat linearen Aufwand und ist abhängig von der Länge des Pfades. Die APs und ACKs werden nur durch das Netz geroutet und müssen, außer von den beiden Kommunikationspartnern, nicht weiter verarbeitet werden.
Des Weiteren ist das Verhalten von quantitativen Größen, wie z.B. die verursachte Netzlast oder die Anzahl der gefundenen Pfade, in Abhängigkeit von der Größe des Netzes interessant. Genau dieses Verhalten wird im nachfolgenden Abschnitt 5.2 mithilfe einer Simulation genauer betrachtet.

Sicherheit

Um die Sicherheit von MuCKE zu diskutieren, werden im Folgenden verschiedene Aspekte betrachtet. Zur Erinnerung: Die Sicherheit von MuCKE stützt sich auf zwei Verfahren: den Multipath-Schlüsselaustausch, der in Kapitel 4 beschrieben wurde, und einen IKEv2-Schlüsselaustausch.

1. **Angreifermodell:** Betrachtet man das im Abschnitt 3.5 beschriebene Angreifermodell, dann stehen dem Angreifer zwei Mittel zur Verfügung:

- Mithilfe eines Quantencomputers ist er in der Lage gängige asymmetrische Kryptosysteme zu brechen.
- Außerdem ist er ein berechtigter Teilnehmer im Netz und kann somit Nachrichten senden, empfangen, verwerfen und manipulieren.

Mithilfe des Quantencomputers kann er den durch den IKEv2-Schlüsselaustausch abgeleiteten Schlüssel aus den öffentlichen Informationen berechnen. Der Quantencomputer gibt dem Angreifer aber keinen maßgeblichen Vorteil beim Mutlipath-Schlüsselaustausch. Um das dadurch abgeleitete Geheimnis zu berechnen, muss der Angreifer, unter der Annahme, dass die verwendete KDF sicher ist, alle genutzten Pfade kompromittiert haben. Denn nur dadurch kann er an alle ausgetauschten Zufallszahlen gelangen. Diese Sicherheit hängt also maßgeblich von der Anzahl und Diversität der Pfade ab und wird deshalb im Abschnitt 5.2 mithilfe einer Simulation genauer analysiert.

2. **Verfügbarkeit:** Damit MuCKE für den jeweiligen Knoten (im Folgenden als Alice bezeichnet) nicht mehr verfügbar ist, muss jeder initialisierte Schlüsselaustausch abbrechen. Dazu wird zuerst die Auswirkungen von einem Pollutionangriff diskutiert:
   Sollte ein Angreifer die Daten der Pakete manipulieren, so hat dies Auswirkungen auf das Protokoll. Zuerst wird dabei die Manipulation der EPs und AEPs betrachtet. Mithilfe des EPs wird ein Pfad gefunden und Informationen gesammelt. Der Responderknoten wird diese Informationen speichern und sie mithilfe eines AEPs zurück zum Initiator senden. Dabei werden die Knoten auf diesem Pfad ihre Informationen überprüfen. Sollten sie nicht übereinstimmen, wird das Paket verworfen. Das Manipulieren der EPs kann also nur funktionieren, wenn der Angreifer in der Lage ist, die veränderte Information bei dem dazugehörigen AEPs wieder auf die originale Information zu ändern, bevor das Paket bei dem dazugehörigen Knoten ankommt. Für das Manipulieren der AEPs muss er lediglich darauf achten, dass das AEP bereits bei dem Knoten war, dessen Informationen nun manipuliert werden. Ist es einem Angreifer erfolgreich gelungen, die Informationen so zu manipulieren, dass die Manipulation nicht aufgefallen ist und der Initiator und der Responder nun verschiedene Informationen besitzen, dann wird die Schlüsselableitung fehlschlagen und die Schlüsselaushandlung wird abgebrochen. Sollte der Angreifer die Information des APs, welche Pfade genutzt werden, manipulieren, dann resultiert dies auch in einen unterschiedlichen Schlüssel, was zu einem Abbruch der Schlüsselaushandlung führt. Durch das Manipulieren eines ACKs wird die Information, die zur Überprüfung des Schlüssel genutzt wird, geändert. Überprüft der Initiator nun den Schlüssel, dann schlägt dies fehl und die Schlüsselaushandlung wird abgebrochen.
   Der Angreifer kann also die Schlüsselaushandlung stören, sollte er erfolgreich die EPs, AEPs, APs oder ACKs manipulieren. Für die EPs und AEPs ist dies nicht trivial, da die Knoten auf dem Rückweg die Informationen überprüfen. Gelingt es dem Angreifer ein AP oder ACK erfolgreich zu manipulieren und somit die Schlüsselaushandlung zu stören, dann kann er dies wahrscheinlich nicht erneut durchführen, falls die Schlüsselaushandlung wiederholt wird. Die APs und ACKs werden über einen zufälligen Pfad geroutet. Die Wahrscheinlichkeit, dass solch ein Paket zwei Mal über denselben Pfad geroutet wird, ist $|\mathfrak{W}|^{-2}$, wobei $\mathfrak{W}$ die Menge der Pfade ist. Die Anzahl der genutzten Pfade wird dabei im Abschnitt 5.2 weiter untersucht.

Betrachtet man den Zustandsautomaten in der Abbildung 5.1, dann gibt es drei Möglichkeiten, um in den Zustand „Abbruch“ zu gelangen. Die erste Möglichkeit für einen externen Angreifer ist alle AEPs, die an Alice adressiert sind, zu verwerfen. Dann wird Alice keine AEPs erhalten und somit jeden Schlüsselaustausch verwerfen. Damit dies erfolgreich ist, muss der Angreifer die komplette Kommunikation von Alice überwachen können, was nur möglich ist, wenn er alle Nachbarn von Alice kontrolliert. Die zweite Möglichkeit besteht darin, die APs bzw. ACKs, die von Alice ausgehen bzw. an Alice adressiert sind, zu verwerfen. Auch dann wird Alice den Schlüsselaustausch verwerfen. Da die APs und ACKs über zufällige Pfade geroutet werden, muss der Angreifer auch hier die komplette Kommunikation von Alice kontrollieren, um verlässlich alle Schlüsselaushandlungen zu stören. Die letzte und dritte Möglichkeit besteht darin, die Schlüsselaushandlung zu beeinflussen. Entweder müssen dazu die Pfadinformationen verändert worden sein oder der Nachrichtenaustausch zur Schlüsselüberprüfung wurde gestört. Der erste der beiden Fälle wurde bereits am Anfang dieses Punktes genauer diskutiert. Dafür muss der Angreifer mindestens einen Pfad so kompromittiert haben, dass er die Pfadinformation des dazugehörigen AEPs verändern kann, ohne dass die Knoten auf dem Pfad dies bemerken. Sowohl für den ersten als auch für den zweiten Fall muss der Angreifer die komplette Kommunikation von Alice kontrollieren, um verlässlich die Schlüsselaushandlungen zu stören.
Fasst man dies nun zusammen, dann muss ein Angreifer alle Nachbarn von Alice kompromittieren, um MuCKE verlässlich zu stören und somit das Protokoll nicht verfügbar zu machen. Das Verhalten des Protokolls auf die Störungen eines Angreifers, der nicht alle Nachbarn von Alice kompromittiert hat, sollte weiter quantitativ untersucht werden, um die Abhängigkeit der Ausfallrate von der Anzahl der kompromittierten Knoten zu untersuchen.

3. **Authentisierung:** Das IKEv2-Protokoll führt zwar eine Authentisierung durch, diese ist aber (zum jetzigen Zeitpunkt) nicht quantenresistent. MuCKE bietet dementsprechend, so wie es in Kapitel 4 beschrieben wurde, keine quantenresistente Authentisierung. Diese lässt sich aber mithilfe von hashbasierten Signaturen realisieren. Im Unterabschnitt 2.2.1 wurde die Klasse der hashbasierten Signaturen an dem Beispiel von XMSS vorgestellt. Mithilfe des Standards `XMSS-SHA2_20_256` könnte man eine Authentisierung durchführen. Der größte Nachteil bei diesem Verfahren ist die Zustandsbehaftung. Es können (bei diesem Standard) insgesamt 2^{20}-viele Signaturen erzeugt werden, bevor der öffentliche Schlüssel erneuert werden muss. In dem betrachteten Szenario ist dies aber ohne Probleme möglich, da die aufgebauten VPN-Verbindungen für einen längeren Zeitraum aktiv gehalten werden und somit eine Authentisierung nicht häufig vorkommt. Dabei ist das vorgestellte Verfahren nur ein Vorschlag. Das Problem der quantenresistenten Authentisierung im Rahmen des MuCKE-Protokolls sollte weiter untersucht werden.

4. **Einfluss von QKD:** Sollten für die etablierten Verbindungen zur Schlüsselaushandlung QKD-Verfahren genutzt werden, bietet dies für MuCKE einen enormen Vorteil. Mithilfe der QKD-Verfahren können die Schlüssel der einzelnen Verbindungen erneuert werden. Nach so einer Erneuerung kann mithilfe von MuCKE ebenfalls der Schlüssel erneuert werden. Sollte dabei der schlechteste Fall eintreten und dieselben Pfade genutzt werden, steht ein potenzieller Angreifer vor einem neuen Problem. Eventuell kompromittierte Schlüssel bieten ihm keinen Mehrwert

mehr, weswegen er erneut das Netz angreifen muss. Das MuCKE Protokoll in Verbindung mit QKD-Verfahren bietet somit eine sehr hohe Sicherheit.

Robustheit

Die Dynamik des Netzes hat keinen großen Einfluss auf die Funktionalität von MuCKE. Da zu jedem Schlüsselaustausch die Netztopologie mithilfe des Flutens neu erkundet wird, können dadurch dynamische Änderungen, die vor diesem Schlüsselaustausch stattgefunden haben, erkannt werden. Sollte es während des Schlüsselaustauschs zu einer Änderung im Netz kommen, nachdem die EPs die Stelle der Änderung geflutet haben, dann gibt es zwei Möglichkeiten:

- Die Änderung wird erkannt: Dann ist beim Weiterleiten der AEPs eine Änderung des Pfades erkannt worden, wodurch dieser Pfad verworfen wurde.
- Die Änderung wurde nicht erkannt: Dann hatte die Änderung entweder keinen Einfluss auf die gefunden Pfade oder sie ist erst nach dem Weiterleiten der AEPs aufgetreten.

In allen Fällen hat diese Änderung der Netztopologie keine Auswirkung auf die Funktionalität von MuCKE.
Der Verlust von Paketen kann das Protokoll jedoch beeinflussen. Sollten EPs oder AEPs verloren gehen, dann werden dadurch mögliche Pfade nicht gefunden. Dies ist negativ für die Sicherheit, da jeder weitere Pfad eine Verbesserung der Sicherheit bedeutet. Die Funktionalität wäre nur dann beeinflusst, wenn der Initiator innerhalb der `timeWaitForFirstMsg`-Phase kein AEP empfängt. Dann wird der Schlüsselaustausch abgebrochen und es müsste ein neuer Schlüsselaustausch initiiert werden. Sollte ein AP oder ACK verloren gehen, dann kann der Schlüsselaustausch ebenfalls nicht vollendet werden. Das Protokoll versucht dabei aber, die APs zuzustellen und erkennt dies, wenn ein ACK erhalten wird. Sollte dies nicht erfolgreich sein, dann muss der Schlüsselaustausch abgebrochen werden und ein neuer Schlüsselaustausch initiiert werden. Erhält Alice ein ACK, obwohl Bob kein AP erhalten hat, dann wird Alice einen Schlüssel ableiten, Bob nicht. Da nach dem Schlüsselaustausch der abgeleitete Schlüssel mithilfe eines Nachrichtenaustauschs überprüft wird, wird spätestens hier dieser Fehler erkannt. Auch hier müsste ein neuer Schlüsselaustausch initiiert werden.

5.2 Überprüfung quantitativer Anforderungen

In diesem Abschnitt sind der Aufbau einer Testumgebung und die Durchführung verschiedener Messreihen in dieser Testumgebung beschrieben.

5.2.1 Der Testaufbau

Um MuCKE zu testen wurde das Protokoll in das SOLID-System eingepflegt. Mithilfe des SOLID-Systems wird dann ein Netz aufgebaut. Die Netzstruktur entspricht dabei dem VPN-Overlay, welches im Unterabschnitt 2.4.1 beschrieben wurde. Bei jedem Schlüsselaustausch, der bei dem Aufbau des Netzes gestartet wird, wird MuCKE zusätzlich gestartet und leitet somit einen eignen Schlüsselaustausch ein. Alle Parameter werden am Anfang einer Simulation einmal für alle Knoten gesetzt. Insgesamt simuliert dieser Testaufbau den Netzaufbau für 1000 s. Um die Verzögerung zwischen den Netzknoten realistisch zu simulieren, wurden mithilfe von KING [18] reale Verzögerungen gemessen

und für die Simulation genutzt. Um MuCKE zu bewerten, werden folgende Metriken untersucht:

1. die Anzahl der genutzten Pfade eines Schlüsselaustauschs
2. die bewertete Sicherheit eines Schlüssels
3. die Laufzeit eines Schlüsselaustauschs
4. die mittlere Paketgröße

Für die ersten drei Metriken wird dazu jeder Knoten (für die selbst initiierten Schlüsselaushandlungen) diese Informationen speichern. Die Information der Paketgröße speichert sich dabei immer der Knoten, der als Bob agiert. Es werden also auch Pakete von Pfaden gezählt, die eventuell nicht für den Schlüsselaustausch genutzt werden. Über all diese Informationen wird dann pro Metrik ein Mittelwert gebildet. Solch eine Simulation wird 32-mal wiederholt. Zuletzt wird dann über diese 32 Mittelwerte erneut ein Mittelwert gebildet und das 95-Konfidenzintervall berechnet.
Um die Sicherheit zu bewerten wird die in Abschnitt 4.3 beschriebene automatisierte Sicherheitsbewertung genutzt. Dafür wird angenommen, dass jeder Knoten und jede Kante die gleiche Sicherheitswahrscheinlichkeit von 90 % besitzt. Das Verhalten der Metriken soll immer abhängig von der Netzgröße, also der Anzahl der Knoten, betrachtet werden. Das VPN-Overlay ist mithilfe einer Ringstruktur organisiert. Ein Knoten benötigt $\mathcal{O}(\log n)$ viele Suchschritte, um einen anderen Knoten im Netz zu finden. Damit die Auswirkungen des Protokolls vergleichbar sind, wird die Netzgröße exponentiell skaliert.

5.2.2 Testfälle und Messergebnisse

Im Folgenden werden die unterschiedlichen Testfälle beschrieben und ausgewertet. Dazu wird in der ersten Simulation MuCKE, so wie es in Kapitel 4 beschrieben wurde, simuliert. In der zweiten Simulation wird die Anzahl der Pfade begrenzt, um die Laufzeit zu beschleunigen. Dabei wird die Auswirkung dieser Begrenzung auf die anderen Metriken begutachtet.

Simulation 1

Wie bereits im Unterabschnitt 5.2.1 eingeleitet, werden vier verschiedene Metriken betrachtet. Die Laufzeit und die Paketgröße sind aufgrund der im Abschnitt 3.3 gestellten quantitativen Anforderungen an das Protokoll interessant. Zum einem soll die Schlüsselaushandlung nicht länger als 3 s dauern, zum anderen sollen die Pakete nicht größer als 100 B sein.
Die Anforderung der Netzlast lässt sich mithilfe des Parameters `floodAllocation` regulieren. Es können maximal `floodAllocation`-viele EPs nach der Flutphase erzeugt worden sein, die dann zu Bob geroutet werden. Auf jedes EP antwortet Bob mit einem AEP. Zuletzt wird ein Nachrichtenaustausch mithilfe der APs und ACKs ausgeführt, um den Schlüsselaustausch zu beenden. Maximal werden also $2 \cdot \texttt{floodAllocation} + 2$ viele Pakete versendet. Für $\texttt{floodAllocation} \leq 24$ ist die Anforderung, dass maximal 50 Pakete versendet werden sollen, gegeben. Deswegen wird diese Anforderung durch eine Simulation nicht weiter betrachtet.
Um die Qualität des Protokolls zu messen werden die Anzahl der Pfade und die Sicherheit des Schlüssels ermittelt. Dabei hängt die Sicherheit von drei Faktoren ab:

- die Anzahl der Pfade
- die Länge der Pfade
- die Diversität der Pfade

Die Vermutung besteht, dass die Anzahl und die Länge der Pfade mit der Größe des Netzes steigt. Da sich die Anzahl und die Diversität der Pfade positiv und die Länge der Pfade negativ auf die Sicherheit auswirken, ist die Vermutung, dass die Sicherheit bei sehr großen Netzen abnimmt. Die Länge der Pfade dominiert dann die Sicherheitsbewertung. Bei mittleren Netzen sollte die Sicherheit steigen, da die Diversität der Pfade sich verbessern sollte. Um die Sicherheit besser bewerten zu können wird zusätzlich das Verhältnis berechnet, wie viel Schlüsselaushandlungen mit nur einem Pfad realisiert worden sind. Denn diese Schlüsselaushandlungen bieten nur einen minimalen Mehrwert. Für die Simulation wurden folgende Parameter genutzt:

`floodAllocation`	:	20
`maxPeers`	:	5
`timeWaitForFirstMsg`	:	1 s
`timeFlood`	:	1 s

Insgesamt wird das Protokoll in vier verschieden großen Netzen getestet. Die Netzgröße verdoppelt sich dabei in jedem Schritt. Das kleinste Netz besteht aus 25, das größte aus 200 Knoten.

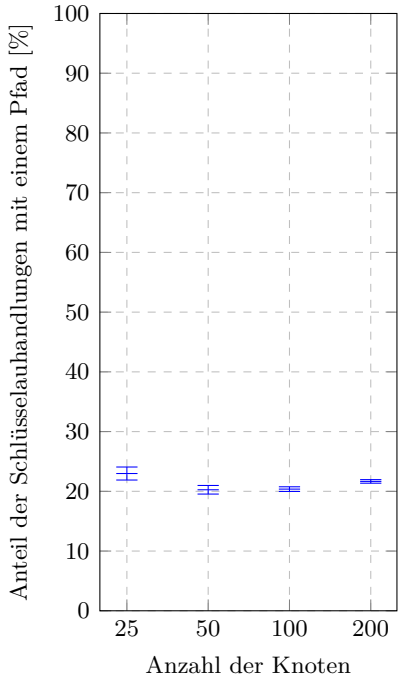
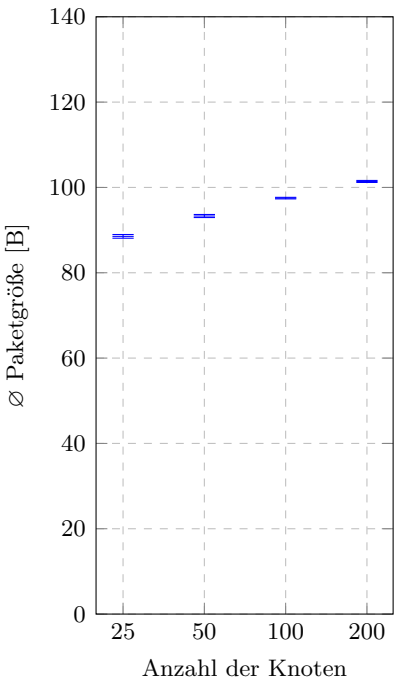
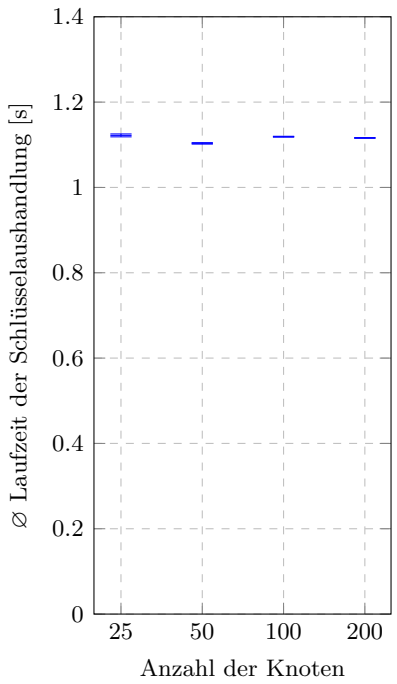


Abbildung 5.2: Prozentsatz der Schlüsselaushandlungen, die nur einen Pfad genutzt haben. Die X-Achse ist logarithmisch skaliert.

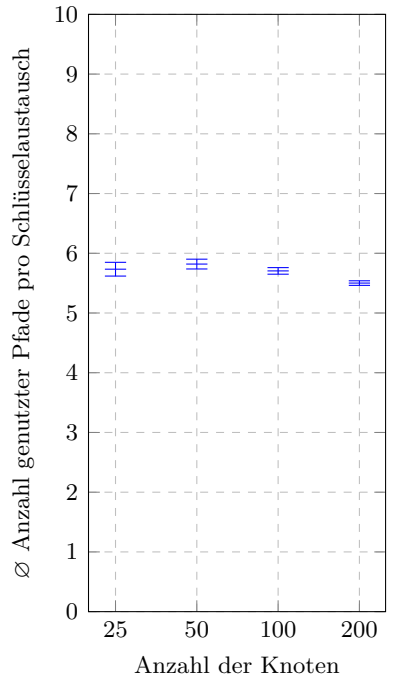
Die Ergebnisse der ersten Simulation sind in der Abbildung 5.3 dargestellt. Die X-Achse ist dabei logarithmisch skaliert. Zuerst betrachte man Abbildung 5.3a. Die Anforderung der Paketgröße kann im Mittel für kleine Netze erfüllt werden. Dabei gibt es aber bereits bei den kleinen Netzen einzelne Pakete, die eine Größe von 100 B überschreiten. Für



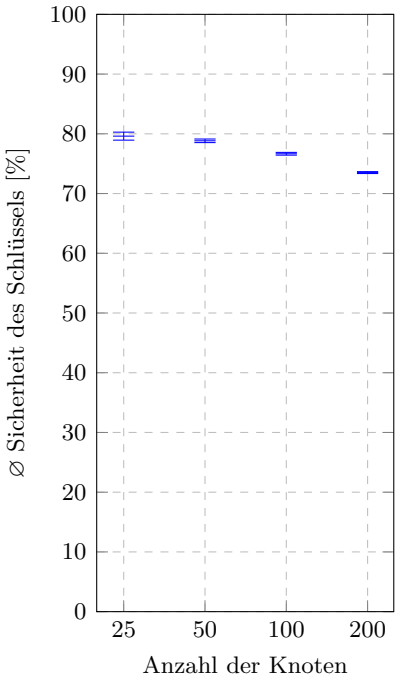
(a) Abhängigkeit der Paketgröße von der Netzgröße



(b) Abhängigkeit der Laufzeit einer Schlüsselaushandlung von der Netzgröße



(c) Abhängigkeit der Anzahl der genutzten Pfade pro Schlüsselaushandlung von der Netzgröße



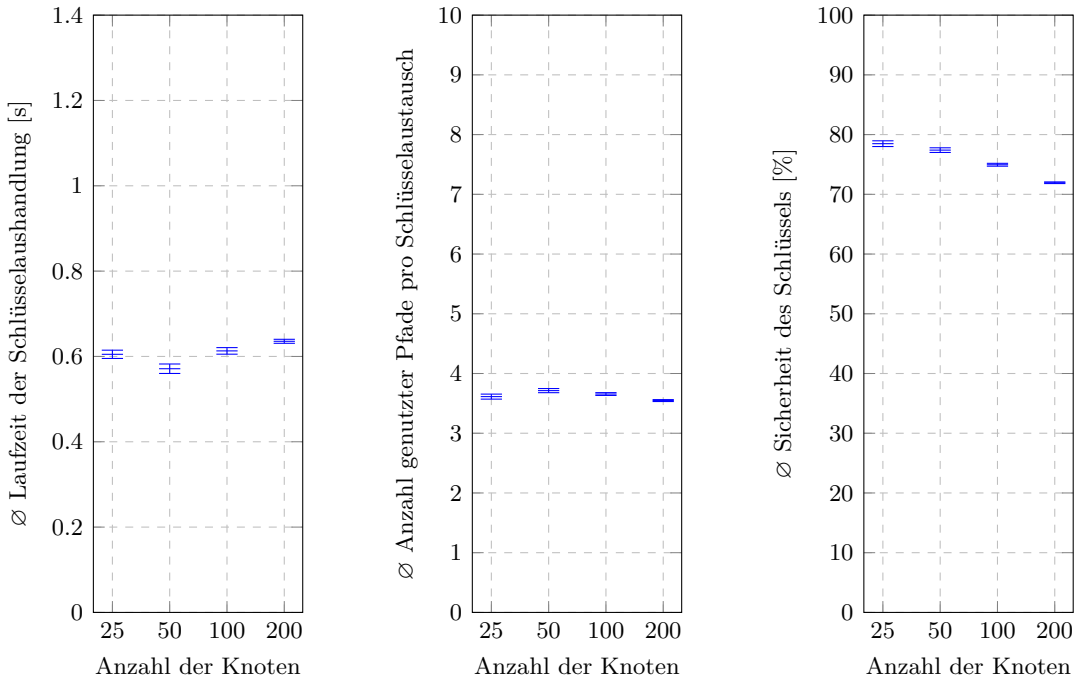
(d) Abhängigkeit der berechneten Sicherheit eines Schlüssels von der Netzgröße

Abbildung 5.3: Ergebnisse der ersten Simulation, die X-Achse ist logarithmisch skaliert.

sehr große Netze ist selbst die mittlere Paketgröße über der gestellten Anforderung. Aus dem Verlauf der Abbildung 5.3a lässt sich vermuten, dass die mittlere Paketgröße logarithmisch von der Größe des Netzes abhängig ist. Dies ist für die Skalierbarkeit wichtig, da so sichergestellt werden kann, dass auch in sehr großen Netzen (> 500 Knoten) die Netzlast nicht überproportional steigt.
Eine weitere gestellte Anforderung ist die Laufzeit, welche in der Abbildung 5.3b dargestellt ist. Die Anforderung der maximale Laufzeit von 3 s wurde dabei unterschritten. Im Schnitt dauert ein Schlüsselaustausch ungefähr 1,15 s. Die Laufzeit ist dabei auch nicht abhängig von der Netzgröße. Auch dies ist eine gute Erkenntnis für die Skalierbarkeit. Auffällig ist, dass die Laufzeit von dem Parameter `timeFlood` dominiert wird. Aus diesem Grund wird in der zweiten Simulation das Protokoll abgeändert, sodass bereits nach fünf gefundenen Pfaden das AP versendet wird. Die Auswertung dieser Simulation wird im Unterunterabschnitt 5.2.2 diskutiert.
In der Abbildung 5.3c ist die mittlere Anzahl der Pfade dargestellt. Entgegen der Vermutung, dass die Anzahl der Pfade mit der Netzgröße korreliert, existiert für ein gegebenes Parameterset eine optimale Netzgröße. Das Maximum wurde bei einem Netz mit 50 Knoten erreicht, wobei die Abweichung zu den anderen Netzen statistisch kaum signifikant ist. Dies sollte auch eine Auswirkung auf die Sicherheit haben, da diese von der Anzahl, Länge und Diversität der Pfade abhängig ist. Die mittlere Anzahl von genutzten Pfaden ist dabei etwas trügerisch. Hingegen der ersten Annahme, dass die Metrik einer Gleichverteilung zugrunde liegt, stellte sich bei genauerer Betrachtung heraus, dass dies nicht der Fall ist. Betrachtet man Abbildung 5.2, dann sieht man, dass ein erheblicher Anteil der Schlüsselaushandlungen nur einen Pfad zum Schlüsselaustausch genutzt hat. Der Grund dafür liegt im Netzaufbau. Zu Beginn hat ein Knoten nur eine einzelne Verbindung, über die er weitere Verbindung aufbaut. Erst wenn diese etabliert sind, kann MuCKE sein volles Potenzial nutzen. Des Weiteren kann man daraus folgern, dass die Anzahl der Pfade sehr große Varianz aufweisen muss, damit ein Mittelwert von ca. 5,75 Pfaden (bei einem Netz mit 25 Knoten) erreicht wird.
In der Abbildung 5.3d wird die mittlere Sicherheit eines ausgehandelten Schlüssels dargestellt. Die getroffene Vermutung, dass die Sicherheit mit der Anzahl der Knoten korreliert, ist gegeben. Dabei verringert sich die Sicherheit mit der Größe des Netzes. Die Länge der Pfade scheint demnach die Sicherheit zu dominieren, weswegen sie für große Netze abnimmt. Trotzdem ist das Ergebnis wünschenswert. Bei einer Sicherheit von 90 % pro Knoten und Kante ergibt sich eine mittlere Sicherheit des Schlüssel von ca. 80 % bei einen kleinem Netz bis hin zu 74 % bei einem Netz mit 200 Knoten. Zum Vergleich: Eine Schlüsselaushandlung über einen Pfad, der aus zwei Knoten und drei Kanten besteht, hat eine Sicherheit von $0{,}9^5 \approx 59\,\%$.

Simulation 2

Bei der zweiten Simulation wurde der gleiche Testaufbau und das gleiche Parameterset wie bei der ersten Simulation genutzt. Der Unterschied zur ersten Simulation besteht im Protokoll. Dazu wurde ein neuer Parameter `maxPaths` eingeführt und auf den Wert fünf gesetzt. Diese Änderung hat folgende Auswirkung: Sobald fünf Pfade gefunden worden sind, wird ein AP versendet und somit die Schlüsselableitung initiiert. Findet man innerhalb der `timeFlood`-Phase keine fünf Pfade, dann wird nach dieser Zeit das AP versendet. Die Vermutung: Die Laufzeit, die durch den `timeFlood`-Parameter dominiert wird, soll sich verbessern. Da aber die Sicherheit mit der Anzahl der Pfade steigt, wird eine Verschlechterung der Sicherheit erwartet. Die Ergebnisse der Messung sind in der Abbildung 5.4 visualisiert.



(a) Abhängigkeit der Laufzeit einer Schlüsselaushandlung von der Netzgröße

(b) Abhängigkeit der Anzahl der genutzten Pfade pro Schlüsselaushandlung von der Netzgröße

(c) Abhängigkeit der berechneten Sicherheit eines Schlüssels von der Netzgröße

Abbildung 5.4: Ergebnisse der zweiten Simulation, bei der bereits nach fünf gefunden Pfaden die Schlüsselableitung initialisiert wurde. Die X-Achse ist logarithmisch skaliert.

Die Begrenzung der Pfade hat eine beträchtliche Auswirkung auf die Laufzeit. Sie konnte dadurch bei dem Netz mit 50 Knoten fast halbiert werden. Vergleicht man Abbildung 5.4a mit der Abbildung 5.2, dann scheint die Laufzeit von dem Anteil der Schlüsselaushandlungen, die nur einen Pfad nutzen, abhängig zu sein. Da bei einer Schlüsselaushandlung über einen Pfad die Änderung keine Auswirkung hat, verschlechtert sich dadurch die Laufzeit.
Die Laufzeitverbesserung ist zwar erfreulich, dabei darf aber die Sicherheit nicht zu stark beeinträchtigt werden. Betrachtet man Abbildung 5.4b, dann ist die mittlere Anzahl genutzter Pfade gesunken. Dies leitet sich aus der Änderung ab, dass nicht mehr als fünf Pfade genutzt werden. Vergleicht man die erreichte Sicherheit aus der ersten Simulation (Abbildung 5.3d) mit der in dieser Simulation erreichten Sicherheit (Abbildung 5.4c), dann ist die Abweichung nur um ca. 2 %, was sehr erfreulich ist. Daraus lässt sich schließen, dass die Diversität der Pfade hoch sein muss, um so eine gute Sicherheit bei maximal fünf Pfaden zu erreichen.

Ausblick: optimale Parameterwahl

Neben der im Unterunterabschnitt 5.2.2 vorgenommenen Änderung am Protokoll, um die Qualität zu verbessern, spielen auch die anderen Parameter eine große Rolle. Das Berechnen optimaler Parameter für ein gegebenes Netz wäre dabei eine weitere gute Verbesserung. Die Time-out Parameter `timeWaitForFirstMsg` und `timeFlood` sollten im besten Fall dynamisch während der Laufzeit berechnet werden. Dieses Optimierungs-

problem ist dabei nicht trivial zu lösen. Es muss dazu zur Laufzeit die durchschnittliche Verzögerung ermittelt werden. Gleichzeitig darf aber ein Angreifer diesen Mechanismus nicht nutzen dürfen, um die Funktionalität des Protokolls zu stören.
Der Parameter `maxPeers` reguliert die maximale Anzahl der Nachbarn. Dabei wird vermutete, dass dieser Parameter mit dem Parameter `floodAllocation` korreliert. Je mehr Pakete man fluten darf, desto mehr Nachbarn können ein Paket erhalten, um eine gleichbleibende „Tiefe“ des Flutens zu erreichen. Beide Parameter sollten dabei abhängig von der Netzgröße und den gestellten Anforderung an die Netzlast gesetzt werden. Auch hier ist die Ermittlung eines optimalen Parameterwerts nicht trivial. Die Auswirkungen dieser Parameter auf die Qualität des Protokolls und die Berechnung optimaler Parameter sollten dabei weiter untersucht werden, um optimale Parametersets bieten zu können.

Zusammenfassung

In diesem Kapitel wurden die im Kapitel 3 formulierten Anforderungen mit den Eigenschaften von MuCKE verglichen. Die Anforderungen wurden dabei erst qualitativ diskutiert. Ausgewählte Anforderungen wurden dann mittels einer Simulation genauer untersucht. Die wichtigsten Ergebnisse sind hier noch einmal zusammengefasst:

- **Skalierbarkeit:** Die Pakete erreichen bei einem Netz mit 200 Knoten im Mittel ein Größe von 101 B. Da die Abweichung zur Anforderung, dass die Pakete nicht größer als 100 B sein sollen, statistisch nicht signifikant ist, kann die Anforderung als erfüllt betrachtet werden. Viel wichtiger ist die Erkenntnis, dass die Paketgröße logarithmisch mit der Größe des Netzes skaliert. Zusätzlich benötigt die Verarbeitung der Pakete höchstens linearen Aufwand. Mithilfe der Simulation konnte eine Verbesserung am Protokoll vorgenommen und die Laufzeit auf ca. 600 ms reduziert werden. Dies zusammen sichert die Skalierbarkeit von MuCKE.

- **Robustheit:** Das Verhalten des Protokolls auf dynamische Änderungen im Netz oder auf den Verlust von Paketen wurde ebenfalls diskutiert. Dynamische Änderungen der Netztopologie haben nur dann einen Einfluss auf MuCKE, wenn bei allen Pfaden eine Änderung auftritt und die AEPs verworfen werden. In allen anderen Fällen ist die Funktionalität nicht gestört. Der Verlust von Paketen kann eine Störung des Protokolls hervorrufen. Für die AP ist dabei ein Zustellungsmechanismus implementiert, welcher die Zustellung sichern soll. Bei dem Verlust von EPs oder AEPs hat nur der Verlust aller EPs oder AEPs eine Auswirkung auf die Funktionalität.

MuCKE kann also die geforderten Anforderung einhalten.

Kapitel 6

Resümee und Ausblick

Das schließende Kapitel fasst zunächst die zentralen Ideen dieser Arbeit zusammen. Im Anschluss wird ein Ausblick auf weitere Entwicklungen am Protokoll MuCKE gegeben.

6.1 Zusammenfassung

Die Entwicklung quantenresistenter Schlüsselaustauschprotokolle ist für die jetzige Zeit eine enorm wichtige Aufgabe. Die Bedrohung der Quantencomputer ist bereits jetzt schon real, da verschlüsselte Daten gespeichert und in Zukunft mithilfe der Quantencomputer entschlüsselt werden können. Der momentane Stand der Technik ist dabei die PQC. Sie bietet vielversprechende quantenresistente asymmetrische Kryptosysteme. Trotzdem sind sie nicht die perfekte Lösung. Entweder haben diese Verfahren eine schlechte Performance oder das Vertrauen in die Sicherheit fehlt.
Lösungen, die auf die PQC verzichten, sind selten. Lediglich der RFC 8784 bietet eine Alternative für WANs, die letztendlich aber nur den IKEv2-Standard dahingehend erweitert, dass in jeder Schlüsselableitung ein PSK mit einbezogen wird. Wie diese PSKs verteilt werden sollen ist nicht weiter spezifiziert.
Deswegen wurde im Rahmen dieser Arbeit MuCKE – ein quantenresistentes Schlüsselaustauschprotokoll – entwickelt, welches auf asymmetrische Verfahren aus der PQC verzichtet. Die Grundlage dafür ist das MKR, welches aus dem Kontext der WSNs stammt. Die Idee: Die zwei Netzknoten befinden sich bereits in einem zusammenhängenden Netz. Man erkundet zwischen diesen beiden Knoten mehrere Pfade und tauscht pro Pfad eine Zufallszahl aus. Die Sicherheit stützt sich dabei auf die bereits existierenden Verbindungen im Netz. Die große Herausforderung dieser Idee in WANs: Wie erkundet man möglichst viele Wege, ohne dabei das Netz zu überlasten und Anforderungen wie Skalierbarkeit oder Robustheit einzuhalten? Die Lösung lieferte das begrenzte Fluten. Es gibt eine maximale Anzahl von gefluteten Paketen, die durch den entwickelnden Algorithmus eingehalten wird.
Zusätzlich zu diesem Schlüsselaustausch bietet MuCKE noch zwei weitere Funktionalitäten. Zum einem kann die erreichte Sicherheit automatisiert bewertet werden. Die dazu benötigten Informationen werden entweder während der Schlüsselaushandlung bereits gesammelt oder können durch Zertifikate bestimmt werden. Zum anderen ist es möglich den Schlüssel mit einem neuen Schlüsselaustausch zu erneuern und somit die Eigenschaft der PFS zu erfüllen. Dabei gibt es zusätzlich einen Fallback-Mechanismus, der die Synchronität wiederherstellen kann, sollte bei dem Erneuern des Schlüssel ein Fehler auftreten.
Das Protokoll wurde dann in das SOLID-System eingepflegt, um es in von SOLID verwalteten Netzen zu evaluieren. Dafür wurde ein Netzaufbau simuliert, um die gestellten Anforderungen mit den Eigenschaften des Protokolls zu vergleichen. Es stellte sich heraus, dass die Anforderungen erfüllt werden. Gleichzeitig wurde bereits während der Evaluation eine Verbesserung am Protokoll vorgenommen, um die Laufzeit bei größeren Netzen zu halbieren.

Letztendlich wurde aus einer einfachen Idee ein umfangreiches Schlüsselaustauschprotokoll, welches die Anforderungen für ein Schlüsselaustauschprotokoll für WANs mit der Quantenresistenz verbindet und zusätzlich auf Verfahren aus der PQC verzichtet.

6.2 Ausblick

Das Protokoll bietet zwar bereits die geforderte Funktionalität der Schlüsselaushandlung. Doch die Qualität des Protokolls lässt sich noch verbessern. Als Erstes sollten weitere Simulationen durchgeführt werden, um die Auswirkungen der verschiedenen Parameter auf die Metriken zu analysieren und schlussendlich Parametersets daraus abzuleiten.
Die Sicherheitsbewertung betrachtet momentan nur die jetzige Situation im Netz. Hier sollte man sich weitere Gedanken machen, wie man den zeitlichen Aspekt mit in die Sicherheitsbewertung einfließen lassen kann. Sollten die Parameter dahingehen geändert werden, dass mehr Pfade gefunden werden, muss eine Approximation für die Berechnung der Sicherheit erfolgen, da über exponentiell viele Möglichkeiten summiert werden muss. Die große Herausforderung hierbei ist, dass diese Möglichkeiten nicht gleichverteilt sind, weswegen eine einfache Monte-Carlo Abschätzung vermutlich zu ungenau ist. Deswegen sollte man die Abschätzung dieser Summe weiter untersuchen.
MuCKE bietet zum jetzigen Zeitpunkt keine Authentisierung der Kommunikationspartner und ist auch nicht in der Lage Zertifikate für die Sicherheitsbewertung zu realisieren. Dafür werden asymmetrische Signaturverfahren aus der PQC benötigt. Auch wenn MuCKE aus genannten Gründen auf die PQC verzichten soll, gibt es bei den hashbasierten Signaturen keine Einwände. Deswegen wurde bereits eine Möglichkeit im Unterabschnitt 5.1.2 vorgeschlagen: XMSS. Jedoch muss der Einsatz von hashbasierten Signaturverfahren in Verbindung mit MuCKE weiter ausgearbeitet und letztendlich evaluiert werden, um die Qualität weiterhin zu sichern. Klar ist, dass durch die Authentisierung die Paketgröße steigt, da zusätzliche Informationen übertragen werden müssen.
Zusammenfassend kann also gesagt werden, dass das in dieser Arbeit entwickelte Protokoll MuCKE ein gute Grundlage liefert, um ein etabliertes Schlüsselaustauschprotokoll für den Einsatzzweck in WANs zu werden.

Anhang A

Literatur

[1] J. Daemen und V. Rijmen, *AES Proposal: Rijndael*, 1999.

[2] R. Rivest, A. Shamir und L. Adleman, „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Jg. 21, S. 120–126, 1978.

[3] P. W. Shor, „Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, Jg. 26, Nr. 5, S. 1484–1509, Okt. 1997, ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. Adresse: http://dx.doi.org/10.1137/S0097539795293172.

[4] F. K. Wilhelm, R. Steinwandt, B. Langenberg u. a., „Entwicklungsstand Quantencomputer," Bundesamt für Sicherheit in der Informationstechnik, BSI Projektnummer 283, 2020, Version 1.2.

[5] L. Meinl, *Die Post-Quanten-Kryptographie. Ein Überblick*. GRIN Verlag, 2021.

[6] H. Chan, A. Perrig und D. Song, „Key Distribution Techniques for Sensor Networks," in *Wireless Sensor Networks*, C. S. Raghavendra, K. M. Sivalingam und T. Znati, Hrsg. Boston, MA: Springer US, 2004, S. 277–303, ISBN: 978-1-4020-7884-2. DOI: 10.1007/978-1-4020-7884-2_13. Adresse: https://doi.org/10.1007/978-1-4020-7884-2_13.

[7] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld und A. Mohaisen, *XMSS: eXtended Merkle Signature Scheme*, RFC 8391, Mai 2018. DOI: 10.17487/RFC8391. Adresse: https://rfc-editor.org/rfc/rfc8391.txt.

[8] R. McEliece, „A public key cryptosystem based on algebraic coding theory," 1978.

[9] H. Singh, „Code based Cryptography: Classic McEliece," *CoRR*, Jg. abs/1907.12754, 2019. arXiv: 1907.12754. Adresse: http://arxiv.org/abs/1907.12754.

[10] J. Hoffstein, J. Pipher und J. H. Silverman, „NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, J. P. Buhler, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, S. 267–288, ISBN: 978-3-540-69113-6.

[11] R. Alléaume, J. Bouda, C. Branciard u. a., „SECOQC White Paper on Quantum Key Distribution and Cryptography," *ArXiv*, Jg. abs/quant-ph/0701168, 2007.

[12] M. Roßberg, „Skalierbare Autokonfiguration sabotageresistenter virtueller privater Netze," Diss., Technische Universität Ilmenau, 2010.

[13] D. Dolev und A. Yao, „On the security of public key protocols," *IEEE Transactions on Information Theory*, Jg. 29, Nr. 2, S. 198–208, 1983. DOI: 10.1109/TIT.1983.1056650.

[14] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen und T. Kivinen, *Internet Key Exchange Protocol Version 2 (IKEv2)*, RFC 7296, Okt. 2014. DOI: 10.17487/RFC7296. Adresse: https://rfc-editor.org/rfc/rfc7296.txt.

[15] C. Tjhai, M. Tomlinson, G. Bartlett u. a., „Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2),“ Internet Engineering Task Force, Internet-Draft draft-tjhai-ipsecme-hybrid-qske-ikev2-02, Juli 2018, Work in Progress, 19 S. Adresse: https://datatracker.ietf.org/doc/html/draft-tjhai-ipsecme-hybrid-qske-ikev2-02.

[16] S. Fluhrer, P. Kampanakis, D. McGrew und V. Smyslov, *Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security*, RFC 8784, Juni 2020. DOI: 10.17487/RFC8784. Adresse: https://rfc-editor.org/rfc/rfc8784.txt.

[17] E. W. Dijkstra, „A note on two problems in connexion with graphs,“ *Numerische Mathematik*, Jg. 1, Nr. 1, S. 269–271, 1959.

[18] K. P. Gummadi, S. Saroiu und S. D. Gribble, „King: Estimating Latency between Arbitrary Internet End Hosts,“ in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, Ser. IMW '02, Marseille, France: Association for Computing Machinery, 2002, S. 5–18, ISBN: 158113603X. DOI: 10.1145/637201.637203. Adresse: https://doi.org/10.1145/637201.637203.

Anhang B

Abkürzungsverzeichnis

AP	Abschluss-Paket
ACK	Acknowledge-Paket
AES	Advanced Encryption Standard
AEP	Antwort-Erkundungspaket
BSI	Bundesamt für Sicherheit in der Informationstechnik
ECC	Elliptische-Kurven-Kryptografie
EP	Erkundungspaket
XOR	exklusives Oder
XMSS	eXtended Merkle Signature Scheme
ID	Identifikationsnummer
IKEv2	Internet Key Exchange Protocol Version 2
KDF	Key Derivation Function
MuCKE	Multipath Cryptographic Key Exchange
MKR	Multipath Key Reinforcement
o.B.d.A.	ohne Beschränkung der Allgemeinheit
PFS	Perfect Forward Secrecy
PQC	Post-Quanten-Kryptographie
PSK	Pre-shared Key
QKD	Quantenschlüsselaustausch
RP	Recovery-Paket
RFC	Request for Comments
RSA	Rivest–Shamir–Adleman
SOLID	Secure OverLay for IPsec Discovery
SPOF	Single Point of Failure
VPN	virtuelles privates Netzwerk
WAN	Weitverkehrsnetz
WSN	Wireless Sensor Network