Post-Quantum Kryptographie für IPsec*

Ephraim Zimmer
Universität Hamburg
FB Informatik, SVS
Vogt-Kölln-Straße 30
22527 Hamburg
ezimmer@informatik.uni-hamburg.de

Mithilfe der *Internet Protocol Security* (IPsec) Suite können wichtige Kommunikationsdaten über IP-basierte Netzwerke auf sichere Art und Weise ausgetauscht werden. Diese Sicherheit stützt sich auf die Annahme, dass der Diffie-Hellman (DH) Schlüsselaustausch als Teil des Internet Key Exchange (IKE) Protokolls, welches unter anderem die kryptographischen Schlüssel für IPsec liefert, nicht gebrochen werden kann. Ein Quantencomputer-gestützter Angriff kann allerdings mithilfe von Shors Algorithmus die kryptographischen Schlüssel berechnen und somit die Sicherheit einer IPsec-Verbindung während und nach dessen Ablauf brechen. Die vorliegende Arbeit beschäftigt sich mit der Integration sogenannter *Post-Quantum Kryptographie* (PQC) in IPsec zum Schutz vor Angriffen durch Quantencomputer. Dabei wurden die durch Quantencomputer gefährdeten Bereiche von IPsec und IKE analysiert und der DH-Schlüsselaustausch im IKE-Protokoll mit einem neuen Schlüsselaustausch basierend auf dem Niederreiter-Verfahren ersetzt. Diese Implementierung wurde anschließend mit aktuell verbreiteten Schlüsselaustauschverfahren verglichen. Die Ergebnisse zeigen neben einer für den praktischen Einsatz hohen Performanz des Niederreiter-Schlüsselaustauschs die erste funktionierende IPsec-Verbindung, die Sicherheit gegen Angriffe durch Quantencomputer bietet. Auf der anderen Seite decken sie allerdings auch die Schwächen des Niederreiter-Verfahrens und die mangelnde Präsenz von PQC in kryptographischen Standards auf.
Die vorliegende Arbeit ist folgendermaßen gegliedert. Kapitel 2 erläutert in Kürze existierende Arbeiten zum vorliegenden Thema. Kapitel 3 analysiert die betrachteten Protokolle sowie deren durch Quantencomputer gefährdete Bereiche. Eine Erläuterung potentieller PQC-Verfahren und die Herausforderungen bei einem praktischen Einsatz erfolgt in Kapitel 5. Auf Grundlage dieser Ausarbeitung wird in Kapitel 6 eines dieser Verfahren ausgewählt, prototypisch in IPsec integriert und in Kapitel 7 dessen Probleme und Lösungsansätze erläutert. Abschließend wird diese praktische Umsetzung in Kapitel 8 evaluiert und in Kapitel 9 auf weitere Forschungsmöglichkeiten hingewiesen.

* Dieses Paper basiert auf der Diplomarbeit [34] des Autors.

1 Einleitung

Die Protokoll-Suite IPsec bietet zur Sicherung wichtiger Kommunikation über IP-basierte Netzwerke unter anderem die drei wesentlichen Sicherheitsdienste Teilnehmerauthentifizierung, Vertraulichkeit und Integritätsschutz der digitalen Kommunikation [17]. Dabei stützt sich IPsec auf die automatisierte Verteilung und Verwaltung kryptographischer Schlüssel durch das IKE Protokoll sowie auf kryptographische Algorithmen, die nach heutigem Stand der Wissenschaft und Technik grundsätzlich ein hohes Maß an Sicherheit bieten, jedoch gegen einen Quantencomputer-gestützten Angriff Schwächen aufweisen [5].

Bereits 1994 veröffentlichte PETER SHOR in seinem Artikel „Algorithms for Quantum Computation: Discrete Logarithms and Factoring“ [29] einen Algorithmus, der, ausgeführt auf einem universellen Quantencomputer, das Brechen aller auf diesen beiden zahlentheoretischen Problemen basierenden Sicherheitssysteme ermöglicht. Zwei Jahre später entwickelte LOV GROVER einen Quantenalgorithmus, mit dem ein unsortierter Datensatz, beispielsweise der Schlüsselraum eines symmetrischen kryptographischen Schlüssels, quadratisch schneller durchsucht werden kann als mit bekannten und als optimal angesehenen klassischen Algorithmen [12].

IPsec setzt zum Schutz der Vertraulichkeit und Integrität symmetrische kryptographische Algorithmen und Hashverfahren ein, die durch Grovers Algorithmus angreifbar sind. Da dieser allerdings nur einen quadratischen Geschwindigkeitsgewinn erbringt, kann dessen Auswirkung auf kryptographische Verfahren durch eine Verdoppelung der Schlüssellängen kompensiert werden. Problematischer verhält es sich mit dem IKE-Protokoll, welches die Grundlage für alle Sicherheitsdienste und Schutzziele von IPsec liefert. Das IKE-Protokoll verwendet zum sicheren Austausch kryptographischer Schlüssel, die in IPsec zum Einsatz kommen, den DH Schlüsselaustausch sowie digitale Zertifikate zur Authentifizierung der Kommunikationspartner [16]. Die Sicherheit der dabei zugrunde liegenden kryptographischen Verfahren basieren auf der Annahme, dass das Faktorisieren großer Integerzahlen und das Finden des diskreten Logarithmus durch keinen Algorithmus in polynomieller Zeit durchgeführt werden kann. Die Anwendung von Shors Algorithmus bedeutet das Brechen dieser Verfahren. Um die drohende Gefahr durch Quantencomputer abzuwenden, konzentriert sich das Wissenschaftsfeld der PQC auf die Suche nach kryptographischen Verfahren, die nicht durch Quantenalgorithmen angreifbar sind [5]. Zusätzlich verfolgt es das Ziel der Optimierung vorgeschlagener PQC-Kandidaten hinsichtlich der Performanz und des Speicherverbrauchs sowie das Ziel der Erhöhung der praktischen Benutzbarkeit. Zu diesem Zweck fehlt es allerdings sowohl in der Wissenschaft als auch in der Praxis an praktischen Umsetzungen und belastbaren Evaluationen von PQC-Verfahren in realen kryptographiegestützten Kommunikationssystemen.

Die vorliegende Arbeit versucht diese Lücke zu füllen und beschäftigt sich umfassend mit der Integration von PQC in IPsec. Dabei werden die folgenden Forschungsbeiträge geliefert:

- Vollständige Analyse der an einer IPsec-Sitzung beteiligten Protokolle sowie deren durch Quantencomputer gefährdeten Bereiche,
- Erläuterung potentieller PQC-Verfahren für die Verwendung eines Schlüsselaustauschs und deren Herausforderungen bei einem praktischen Einsatz,

- Prototypische Implementierung eines ausgewählten PQC-Schlüsselaustauschverfahrens und praktische Integration in ein IPsec-basiertes Kommunikationssystem,
- Simulation, Analyse und Evaluierung des praktischen Einsatzes im Vergleich mit aktuell verwendeten Kryptoverfahren.

2 Existierende Arbeiten

Das Bundesamt für Sicherheit im Internet (BSI) und die Firma secunet beschäftigen sich bereits seit einigen Jahren mit dem Thema PQC, speziell für die secunet-eigene Sichere Inter-Netzwerk Architektur (SINA)-Produktlinie, die auf der unter dem Namen *strongSwan* bekannt gewordenen IKE-Implementierung aufbaut. Dabei ist im Jahr 2011 ein nicht-öffentliches Grobkonzept zum Thema „Integration von Quantencomputer-sicheren Kryptoverfahren im System SINA“ entwickelt worden. Darin wird beschrieben welche Schritte unternommen werden müssen, um ein ausgewähltes, als Quantencomputer-resistent eingeschätztes, Kryptoverfahren als Schlüsselaustauschprotokoll in SINA bzw. IKEv1 und IKEv2 zu integrieren. Bei dem ausgewählten PQC-Verfahren handelt es sich um das codebasierte Verschlüsselungssystem von NIEDERREITER. Eine genaue Begründung für diese Wahl ist in dem Grobkonzept allerdings nicht enthalten.

Der Versuch von [30] ein PQC-Verfahren namens NTRU[1] mit einem RFC Entwurf in den Standard des kryptographischen Protokolls TLS zu integrieren, welches ähnliche Sicherheitsdienste bereitstellt wie IPsec, wurde seit 2001 nicht weiter verfolgt und der Entwurf verlor damit seine Gültigkeit.

Das strongSwan Projekt gab Ende November 2013 bekannt, dass ab der Version 5.1.2, die im Februar 2014 veröffentlicht wurde [32], die Einführung des NTRU-Verfahrens geplant sei.[2] Aufgrund des Fortschritts der vorliegenden Arbeit zum Zeitpunkt der Bekanntgabe konnten allerdings keine Erkenntnisse aus dieser Entwicklung in die praktische Umsetzung einfließen.

3 Analyse von IPsec

Für die Umsetzung der Schutzziele und Sicherheitsdienste von IPsec werden hauptsächlich das Konzept der Security Associations (SAs) sowie je nach Konfiguration die Protokolle Encapsulating Security Payload (ESP) und Authentication Header (AH) eingesetzt, die wiederum spezielle kryptographische Algorithmen anwenden. Diese Protokolle müssen zum Schutz vor Angriffen durch Quantencomputer umfassend analysiert und deren gefährdete Bereiche identifiziert werden. Eine gesonderte Betrachtung des IKE-Protokolls ist besonders wichtig, da alle IPsec-Sicherheitsdienste auf dem durch IKE durchgeführten sicheren und authentifizierten Schlüsselaustausch basieren.

[1] Vergleiche Abschnitt 5.2 für eine kurze Erläuterung des NTRU-Verfahrens.

[2] Genaue Informationen über die Bekanntgabe sind unter `https://github.com/strongswan/strongswan/commit/acc25f29bdf5e2e8e919041e23151877f158dc9d` zu finden.

3.1 SA

Eine SA legt fest, auf welche Art und Weise eine Kommunikationsbeziehung gesichert ist. Sie enthält dafür sicherheitsrelevante Aspekte, wie einzusetzende kryptographische Algorithmen und Schlüssel. Um die durch eine SA festgelegten Sicherheitsmechanismen im Kontext von IPsec umzusetzen, zeigen die eingesetzten Protokolle AH und ESP jeweils mithilfe des übertragenen Security Parameter Index (SPI) in ihrem Header auf die ausgehandelten SAs. Sie weisen somit auf die nötigen Sicherheitsinformationen hin, mit denen z. B. ein empfangenes Paket korrekt entschlüsselt bzw. auf Integrität geprüft werden kann.

3.2 AH und ESP

Mithilfe einer solchen zuvor zwischen den Teilnehmern ausgehandelten SA und den darin enthaltenen kryptographischen Schlüsseln wird mit AH ein schlüsselbasierter Message Authentication Code (MAC) berechnet, in den der Payload eines IP-Paketes und der größtmögliche Teil des ursprünglichen IP-Headers – ausgenommen von Teilen des IP-Headers, die legitimen Veränderungen unterliegen[3] – einfließen. Diese Information wird zur Authentifizierung des Senders und zum Schutz der Integrität jedem IP-Paket als Teil eines neuen Headers hinzugefügt.

Das ESP-Protokoll fügt neben einem ESP-Header noch einen ESP-Trailer in das IP-Paket ein und stellt im Gegensatz zu AH optional die Vertraulichkeit der Kommunikationsdaten bereit. Dies kann erreicht werden, indem der ESP-Payload sowie der ESP-Trailer mithilfe der kryptographischen Schlüssel einer zuvor ausgehandelten SA verschlüsselt werden. Zusätzlich kann wiederum zur Senderauthentifizierung und dem Integritätsschutz ein schlüsselbasierter MAC über den ESP-Header, Payload und Trailer errechnet und im ESP-Authentifizierungsdatenfeld (ICV) an den ESP-Trailer angehängt werden.

3.3 IKE

Der initiale Schlüsselaustausch und die Aushandlung von SAs für AH oder ESP zwischen zwei Kommunikationspartnern sind die Basis für alle von IPsec bereitgestellten Schutzziele und Sicherheitsdienste. Demnach ist dieser Vorgang kritisch für die Sicherheit der Kommunikation und muss somit genauer zu betrachtet werden.

IKE ist das primär eingesetzte Protokoll für diese beiden Aufgaben, da es diese Vorgänge automatisiert und dynamisch durchführt. Dies entlastet die Kommunikationspartner vom persönlichen Austausch sowie der manuellen Konfiguration dieser Parameter und skaliert somit besonders gut bei mehreren und weit entfernten Teilnehmern.

Nachfolgend wird der Ablauf einer IKE-Sitzung ohne Fehler und Ausnahmen zwischen zwei Kommunikationsteilnehmern *Alice* und *Bob* beschrieben. Die Bedeutung der Abkürzungen

[3] Diese Veränderungen sind legitim und betreffen z. B. das Dekrementieren des Time-To-Live Feldes und die sich damit ebenfalls ändernde IP-Header Prüfsumme.

Notation	Payload
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
HDR	IKE header (not a payload)
IDi, IDr	Identification - Initiator / Responder
KE	Key Exchange
SA	Security Association
SK	Encrypted and Authenticated
TS_i, TS_r	Traffic Selector - Initiator / Responder

Tabelle 1: Die Abkürzungen der Payloads der IKE-Nachrichten. In Anlehnung an [16].

der Payloads einzelner Nachrichten können Tabelle 1 entnommen werden. Eckige Klammern weisen auf optional einsetzbare Payloads hin, und die Schreibweise $SK\{\ldots\}$ bezeichnet Daten innerhalb der geschweiften Klammern, die verschlüsselt und integritätsgeschützt sind.

IKE_SA_INIT

```
   Alice                                Bob
   ------------------------------------------------------------------
1.  HDR, SAi1, KEi, Ni  -->
2.                                      <--  HDR, SAr1, KEr, Nr, [CERTREQ]
```

Listing 1: Das erste IKEv2 Nachrichtenpaar IKE_SA_INIT. Quelle: [16].

Zunächst wird das als IKE_SA_INIT bezeichnete Nachrichtenpaar gesendet. Darin schlägt Alice als Initiator der IKE-Sitzung mit SA_{i1} eine Menge von unterstützten kryptographischen Algorithmen und deren Parameter für die IKE SA vor. Darin enthalten müssen Vorschläge für Authentifizierungs- und Verschlüsselungsalgorithmen sowie für einen DH-Algorithmus sein. Weiterhin übermittelt Alice mit KE_i ihren Teil für den DH-Schlüsselaustausch sowie eine einmalig verwendete zufällige Nummer N_i, genannt Nonce.[4]

Der Empfänger Bob sendet seine Auswahl der zu verwendenden kryptographischen Algorithmen und seinen Teil für den DH Schlüsselaustausch sowie eine eigene Nonce an Alice zurück. Zusätzlich kann er die Verwendung eines Zertifikates für die nachfolgende Authentifizierung verlangen, indem er mit dem CERTREQ eine Liste der für ihn vertrauenswürdigen Zertifikatsauthoritäten (CA, engl. Certificate Authority) sendet.

Das Resultat der IKE_SA_INIT ist die gemeinsame IKE SA, inklusive aller benötigten kryptographischen Schlüssel, die auf Grundlage des aus dem DH Schlüsselaustausch hervorgegangenen gemeinsamen Geheimnisses berechnet wurden. Diese IKE SA wird daraufhin

[4] Nonce ist eine Zusammensetzung der Wörter *number used once* und wird häufig im kryptographischen Kontext für zufällige oder pseudo-zufällige Zahlen gebraucht, die nur einmal verwendet werden dürfen.

verwendet, um alle nachfolgenden Nachrichten durch Verschlüsselung und Integritätsschutz zu sichern.

Eine Teilnehmerauthentifizierung ist an dieser Stelle noch nicht vorgenommen, was prinzipiell die Möglichkeit eines Man-in-the-Middle Angriffs ermöglicht. Auch die Integrität der ausgetauschten Nachrichten wurde noch nicht gesichert. Aus diesem Grund ist der darauffolgende Nachrichtenaustausch IKE_AUTH unbedingt nötig für die Sicherheit der gerade ausgetauschten IKE SA.

IKE_AUTH

```
   Alice                              Bob
   ------------------------------------------------------------------
3.  HDR, SK{IDi, [CERT,] [CERTREQ,]
     [IDr,] AUTH, SAi2,
     TSi, TSr}          -->
4.                                    <--  HDR, SK{IDr, [CERT,] AUTH,
                                           SAr2, TSi, TSr}
```

Listing 2: Das zweite IKEv2 Nachrichtenpaar IKE_AUTH. Quelle: [16].

Mit dem als IKE_AUTH bezeichneten Nachrichtenpaar werden die beiden ersten Nachrichten des IKE_SA_INIT authentifiziert und die erste Child SA wird ausgehandelt.

Dafür sendet Alice ein identifizierendes Merkmal mit ID_i[5] und den Integritätsschutz sowie die Authentifizierung der ersten im IKE_SA_INIT Schritt gesendeten Nachricht mit dem AUTH-Payload. Die Authentifizierung erfolgt durch die Berechnung einer digitalen Signatur oder eines MACs über die wichtigsten Teile der ersten Nachricht. Hinzugefügt wird darüber hinaus ein Teil des errechneten kryptographischen Schlüssels, der aus dem DH-Schlüsselaustausch der IKE_SA_INIT hervorging und das gerade gesendete Merkmal ID_i.

Die Verwendung einer digitalen Signatur oder eines MACs hängt davon ab, ob Bob im IKE_SA_INIT Schritt ein digitales Zertifikat angefordert hat und ob beide Kommunikationsteilnehmer vor Beginn der Kommunikation auf anderem Wege einen gemeinsamen Schlüssel (engl. pre-shared key) ausgetauscht haben,[6] der für den MAC benötigt wird.

Weiterhin enthält diese Nachricht mit SA_{i2} wieder eine Menge der unterstützen kryptographischen Algorithmen sowie deren Parameter, diesmal für die erste Child SA. Der Traffic Selector TS_i (bzw. TS_r) spielt für die vorliegende Arbeit keine Rolle.

Sofern Bob im IKE_SA_INIT ein Zertifikat von Alice angefordert hat, enthält diese Nachricht das gewünschte digitale Zertifikat von Alice, mit dessen öffentlichem Schlüssel die Nachrichtensignatur im AUTH von Bob geprüft werden kann. Zusätzlich kann auch Alice mit

[5] Dieses identifizierende Merkmal könnte beispielsweise die eigene IP-Adresse sein, sofern dies ausreichend ist.

[6] Beispielsweise kann der Administrator der IPsec-Sitzung diesen pre-shared key auf beiden IPsec-Endgeräten vorkonfigurieren.

CERTREQ ein Zertifikat von Bob anfordern.

Bob prüft die Authentifizierung von Alice und sendet seinerseits diese Informationen an sie zurück, mit dem Unterschied, dass er die zu verwendenden kryptographischen Algorithmen für die erste Child SA aus der vorgeschlagenen Menge auswählt und seine Festlegung zurück sendet. Das Resultat bei erfolgreichem IKE_AUTH ist die gegenseitige Authentifizierung der Identitäten und aller bis dahin gesendeten Nachrichten. Weiterhin kann die erste Child SA von ESP oder AH verwendet werden.

Alle Payloads mit Ausnahme des IKE-Headers sind verschlüsselt und deren Integrität ist geschützt durch den Einsatz der IKE SA.

CREATE_CHILD_SA

```
   Alice                                Bob
   ------------------------------------------------------------------
5.  HDR, SK{SA, Ni, [KEi],
     TSi, TSr}          -->
6.                                      <--  HDR, SK{SA, Nr, [KEr],
                                               TSi, TSr}
```

Listing 3: Weitere IKEv2-Nachrichtenpaare CREATE_CHILD_SA. Quelle: [16].

Durch CREATE_CHILD_SA können beliebig viele weitere SAs für ESP oder AH zwischen Alice und Bob erstellt werden. Der Ablauf ist ähnlich wie der im IKE_SA_INIT, mit dem bedeutenden Unterschied, dass hier alle Payloads bis auf den IKE-Header verschlüsselt und integritätsgeschützt sind.

Alice schlägt mit SA wieder eine Menge der unterstützen kryptographischen Algorithmen vor, sendet eine neue Nonce N_i und optional neues Schlüsselmaterial KE_i für einen DH-Schlüsselaustausch. Das neue KE_i ist prinzipiell nicht nötig, da beide Teilnehmer das bereits ausgehandelte Schlüsselmaterial der IKE_SA weiter verwenden können. Möchte man allerdings sogenannte *Perfect Forward Secrecy* (PFS) erreichen, so müssen beide Teilnehmer für jede neue CHILD_SA neues Schlüsselmaterial austauschen. Mehr zu diesem Thema im Kontext von IKE ist in [16] zu finden.

Bob sendet seine Festlegung auf eine SA, ebenfalls eine neue Nonce N_r und seinen Teil KE_r für den neuen DH-Schlüsselaustausch, falls angefordert.

3.4 Sicherheitsanalyse

Die theoretische Sicherheit[7] einer IPsec-geschützten Kommunikation ist abhängig von den eingesetzten kryptographischen Protokollen, der Verteilung und Stärke der kryptographischen

[7] Für die praktische Sicherheit ist zusätzlich die Betrachtung der tatsächlichen Implementierung notwendig.

Schlüssel und der zugrunde gelegten Angreifermodelle. Diese drei Kategorien werden im Folgenden analysiert.

Angreifermodelle

Ein Angreifermodell ist laut [27] die maximal berücksichtigte Stärke eines Angreifers und wird verwendet, um den Schutz der Teilnehmer eines Systems festzustellen. Dabei unterscheidet PFITZMANN die Charakteristiken *Rolle, Verbreitung, Verhalten, Rechenkapazität und Rechenzeit* des Angreifers, die jeweils näher eingegrenzt und zusammen genommen das Angreifermodell darstellen.

Abhängig vom Kontext kann IPsec prinzipiell Schutz gegen jegliches Angreifermodell bieten, das die Rollen Außenstehender und Benutzer umfasst. Deren Verbreitung muss allerdings genügend beschränkt sein, um keine Denial-of-Service Angriffe durchführen zu können und um keinen Zugriff auf eines oder mehrere IPsec-Endgeräte[8] der Kommunikation zu besitzen.[9] Durch persönlichen Austausch ausreichend vieler und langer Schlüssel und durch die Verwendung informationstheoretisch sicherer Kryptoalgorithmen könnten diese Angreifermodelle ausgeschlossen werden.

Betreiber, Wartungsdienst, Produzent und Entwerfer eines IPsec-Systems können Hintertüren und Lücken in die praktische Umsetzung des System einarbeiten oder entsprechende Konfigurationen vornehmen, die einen Angriff ermöglichen. Demnach kann vor Angreifermodellen mit diesen Rollen durch IPsec allein nicht geschützt werden.

Für den praktischen Einsatz von IPsec werden zusätzlich die nachfolgend erläuterten kryptographische Protokolle vorgeschlagen und eingesetzt, die keinen Schutz gegenüber komplexitätstheoretisch unbeschränkten Angreifern bieten. Dies ist ein Kompromiss zwischen Sicherheit und Nutzbarkeit. Denn andernfalls müssten die Kommunikationspartner informationstheoretisch sichere Kryptoalgorithmen einsetzten und für jede einzelne Nachricht neue zufällige Schlüssel auf informationstheoretisch sichere Art und Weise austauschen, die zudem noch genauso lang sein müssen wie die Nachricht selbst.[10]

Somit werden IPsec die folgenden Angreifermodelle zugrunde gelegt, welche im Verlauf der vorliegenden Arbeit als Referenz bei der Evaluation der Änderungen an IKEv2 dienen sollen:

[8] Mit IPsec-Endgerät ist das Gerät gemeint, an dem die durch IPsec geschützte Kommunikation beginnt oder endet. Das kann zum einen mit den Endgeräten der Kommunikationsteilnehmer zusammen fallen. Zum anderen kann es sich dabei allerdings auch um Netzwerkknoten wie Router oder spezielle IPsec-Gateways handeln, die ein Subnetz hinter sich verbergen.

[9] Hätte ein Angreifer Zugriff auf wenigstens ein IPsec-Endgerät der Kommunikationsbeziehung, so hätte er einerseits Zugang zu den kryptographischen Schlüsseln und andererseits könnte er IPsec so konfigurieren, dass ein Brechen gewünschter Schutzziele effektiv durchführbar wäre oder die Verbindung zum Kommunikationspartner nicht zustande kommt.

[10] Vergleiche [27].

Angreifermodell 1

Ein Angreifer habe die Rolle eines Außenstehenden.
Als dieser habe er nur Zugriff auf die Subsysteme zwischen den beiden IPsec-Endgeräten der Kommunikationsteilnehmer, um durch IPsec gesicherte Nachrichten abzufangen. Er verfügt allerdings nicht über eine genügend große Verbreitung zur effektiven Unterbindung der IPsec-Kommunikation.
Sein Verhalten sei als aktiv und modifizierend charakterisiert. Das bedeutet der Angreifer kann Nachrichten an einen der Kommunikationsteilnehmer senden, die darauf hin von diesem unverändert über eine durch IPsec geschützte Kommunikationsverbindung an den zweiten Teilnehmer gesendet werden. Darüber hinaus kann er für seine Rolle unerlaubte Modifikationen an übermittelten IPsec-Paketen durchführen.
Die verfügbare Rechenkapazität und Zeit des Angreifers sei komplexitätstheoretisch beschränkt.

Angreifermodell 2

Ein Angreifer habe alle Rollen und Charakteristiken des Angreifermodells 1 und verfüge zusätzlich über einen Quantencomputer, auf dem er in Echtzeit Shors und Grovers Algorithmen ausführen kann.

kryptographische Protokolle

In den Dokumenten RFC7321 und RFC4307 sind aktuell die kryptographischen Algorithmen benannt, die von ESP, AH und IKE unterstützt werden müssen [23, 28], um ihrer Spezifikation zu entsprechen. Es gibt noch weitere Algorithmen, die von IPsec eingesetzt werden können.[11] Im Folgenden werden allerdings nur die verpflichtend zu implementierenden Algorithmen analysiert, die in Tabelle 2 aufgelistet sind.

Zur Senderauthentifizierung und dem Schutz der Integrität von Nachrichten müssen sowohl die Implementierungen von ESP als auch von AH und von IKE den Algorithmus **HMAC-SHA1-96** unterstützen. Dabei handelt es sich um einen MAC, der mithilfe einer kryptographischen Hashfunktion und einem geheimen Schlüssel – in diesem Fall sind dies SHA1 und der geheime Schlüssel zur Authentifizierung zwischen den Kommunikationsteilnehmern – über einer Nachricht errechnet wird. Gesendet werden von der 160 Bit Ausgabe, die der Algorithmus produziert, nur die ersten 96 Bit.[12] Mit einem Schlüssel K und zwei vorgegebenen Konstanten $opad$ und $ipad$ wird der HMAC-SHA1 einer Nachricht M folgendermaßen berechnet:

$$\begin{aligned}\text{HMAC}-\text{SHA1}_K(M) = \text{SHA1}((K \oplus opad)\,||\\ \text{SHA1}((K \oplus ipad)\,||\,M)),\end{aligned}$$

wobei $\oplus$ die Addition modulo 2 (XOR) und $||$ die Konkatenation bezeichnet.

[11] Weitere Standard-konforme Algorithmen sind unter [14] und [15] zu finden.

[12] Eine Erklärung für die Verkürzung des HMAC findet sich in [18].

	Algorithmus	Bit-Sicherheit
Authentifizierung	HMAC-SHA1-96	160
Verschlüsselung	AES-CBC-128	128
Schlüsselverteilung	DH-1024 MODP	80

Tabelle 2: Die laut RFC7321 und RFC4307 von ESP, AH und IKE als verpflichtend zu implementierenden Algorithmen und deren Bit-Sicherheit.

Die Sicherheit des HMAC-SHA1-96 Algorithmus hängt, neben der Qualität und Länge des verwendeten geheimen Schlüssels, hauptsächlich von der Stärke des HMAC-Algorithmus und der Kollisionsresistenz der verwendeten Hashfunktion SHA1 ab, wobei Sicherheitslücken im SHA1 einen kleineren Einfluss auf die Sicherheit des HMAC-SHA1 Algorithmus hätten als auf SHA1 selbst [18]. Der häufigste Angriff ist somit laut [31] die vollständige Suche des geheimen Schlüssels, der in diesem Fall eine Länge von 160 Bit – der Blocklänge der Hashfunktion – haben muss. Daraus resultiert eine 160 Bit-Sicherheit des HMAC-SHA1-96 Algorithmus.

Zusätzlich wird dieser Algorithmus als Pseudozufallsfunktion (PRF) zur Erzeugung langer kryptographischer Schlüssel verwendet, die als Startwert ein gemeinsames Geheimnis der Kommunikationsteilnehmer verwendet.

Für die Verschlüsselung und damit zur Erreichung der Vertraulichkeit ist der Algorithmus **AES-CBC-128** von ESP und IKE verpflichtend zu unterstützen. Dabei wird der Advanced Encryption Standard (AES)-Algorithmus, ein vom National Institute of Standards and Technology (NIST) standardisierter symmetrischer Blockalgorithmus, mit geheimen Schlüsseln der Länge 128 Bit im sogenannten Cipher Block Chaining (CBC)-Modus eingesetzt.

Wichtig für die Sicherheit des AES-CBC-128 Algorithmus ist die Qualität und Länge des verwendeten Schlüssels sowie die Sicherheit des CBC-Modus. Der beste bekannte praktische Angriff auf AES allein ist der *Biclique-Angriff* von [8], der die Schlüsselsuche nur geringfügig schneller durchführt als die vollständige Durchsuchung des gesamten Schlüsselraums. Somit kann nach wie vor beim AES-CBC-128 Algorithmus von einer 128 Bit-Sicherheit ausgegangen werden.

Mit dem CBC-Modus wird jeder Nachrichtenblock vor der Verarbeitung (Ver- bzw. Entschlüsselung) mit dem zuvor bearbeiteten Block XOR-addiert. Der erste Block wird mit einem Initialisierungsvektor kombiniert. Der i-te Nachrichtenblock einer Nachricht M wird mit einem Schlüssel K durch AES demnach folgendermaßen verarbeitet:

$$Block_0 = IV, Block_i = \mathrm{AES}(M_i \oplus Block_{i-1}, K).$$

Die Wahl eines speziellen Blockmodus ist deshalb notwendig, da AES ein Blockalgorithmus ist und Nachrichten, die diese Blocklänge überschreiten, nach gewissen Vorgaben in Blöcke aufteilt und verarbeitet. Je nach Wahl des Blockmodus hat dies positive oder negative Auswirkungen auf die Möglichkeit, Nachrichtenblöcke parallel zu verarbeiten sowie auf die semantische Sicherheit des verwendeten Verschlüsselungsalgorithmus.

Der CBC-Modus wird von [3] als sicher im Kontext von *gewählten Klartext Angriffen* (CPA) bezeichnet, erlaubt allerdings nur eine sequenzielle Verarbeitung der Nachrichtenblöcke.

Für die Schlüsselverteilung durch das IKE-Protokoll, die Basis für alle Schutzziele und Sicherheitsdienste von IPsec ist, muss der **DH-1024 MODP** Algorithmus unterstützt werden. Dabei handelt es sich um den DH Algorithmus zum Austausch von Schlüsseln mit einer Modulus-Gruppe der Größe 1024 Bit, dessen Sicherheit auf dem *Diskreten-Logarithmus-Problem* basiert.

Das ausgetauschte Schlüsselmaterial zwischen den Kommunikationsteilnehmern, welches zur Erzeugung des gemeinsamen Geheimnisses verwendet wird, kann durch einen Angreifer errechnet werden, indem er dieses Problem löst, welches laut [31] die gleiche Größenordnung an Komplexität besitzt, wie das Faktorisierungsproblem. Die Berechnung ist somit sowohl zeit- als auch rechenintensiv und hängt maßgeblich von der Größe und Qualität der gewählten Gruppe ab.

Die Bit-Sicherheit asymmetrischer Kryptosysteme ist nicht intuitiv und trivial bestimmbar. Sie hängt laut [19] neben der Effizienz bekannter Angriffe sehr stark von aktueller und zukünftig vorausgesagter Rechentechnik und Kostenfaktoren ab. Das NIST bewertet den DH-1024 MODP Algorithmus mit einer 80 Bit-Sicherheit [**?**].

Ein anderer Algorithmus zur vertraulichen Verteilung von kryptographischen Schlüsseln ist der **Elliptic Curve Diffie-Hellman (ECDH)** Algorithmus. Dabei handelt es sich um eine auf elliptischen Kurven basierende Variante des DH-Algorithmus. Damit verbunden ist das *Diskrete-Logarithmus-Problem über elliptischen Kurven*, was eine höhere Komplexität zum verwandten Problem ohne elliptischen Kurven aufweist und somit kürzere Schlüssellängen bei vergleichbarer Sicherheit bietet [31].

Verteilung der kryptographischen Schlüssel

Unterschieden werden kann die Verteilung von zwei Arten kryptographischer Schlüssel, wobei es im Folgenden irrelevant ist, ob diese Verteilung manuell oder automatisiert abgewickelt wird. Einerseits müssen für IPsec symmetrische Schlüssel, andererseits können auch asymmetrische Schlüssel in Form von digitalen Zertifikaten ausgetauscht werden.

Erfolgt der Schlüsselaustausch der symmetrischen Schlüssel auf unsichere Art und Weise[13] und ein unautorisierter Teilnehmer oder ein Angreifer ist in der Lage, die verteilten kryptographischen Schlüssel in Erfahrung zu bringen, kann er wichtige Sicherheitsdienste von IPsec brechen.

[13] 'Unsicher' in diesem Kontext ist eine sehr dehnbare Bezeichnung und hängt von vielen verschiedenen Faktoren, wie z. B. aktueller Rechentechnik und kryptoanalytischem Fortschritten, aber auch menschlichen-subjektiven Einflüssen, wie z. B. Vorsicht, ab. Je nach dem, ob die Schlüsselverteilung manuell oder automatisiert erfolgt, haben diese Faktoren unterschiedliche Gewichtungen. Gemeint ist ein Schlüsselaustausch, der eine signifikant niedrigere Sicherheit bietet, als der darauffolgende Einsatz dieser Schlüssel in kryptographischen Algorithmen.

Gesendete Nachrichten können abgefangen und mithilfe der in Erfahrung gebrachten Schlüssel entschlüsselt sowie beliebig verändert und weitergesendet werden. Die Sicherheitsdienste Vertraulichkeit, verbindungslose Integrität, eingeschränkte Datenfluss-Vertraulichkeit und der optionale Replay-Schutz sind damit gebrochen.

Werden die symmetrischen Schlüssel und digitalen Zertifikate zusätzlich nicht authentifiziert ausgetauscht, sind sogenannte Man-in-the-Middle Angriffe[14] möglich. Die beiden verbleibenden Sicherheitsdienste Senderauthentifzierung und damit auch die Zugriffskontrolle können somit nicht mehr durch IPsec bereitgestellt werden.

4 Integration von Post-Quantum Kryptographie in IPsec

IPsec muss nicht von Grund auf neu gestaltet werden um Resistenz gegenüber Quantencomputern zu erreichen. Nachfolgend werden die nötigen Änderungen aufgezeigt, die an den Protokollen ESP, AH und IKE vorgenommen werden müssen, um PQC in IPsec zu integrieren.

4.1 Änderungen an ESP und AH

Die Analyse von ESP und AH in Abschnitt 3 hat gezeigt, dass beide Protokolle ihre Sicherheitsdienste ausschließlich durch den Einsatz von einerseits hashbasierten und andererseits symmetrischen Kryptoverfahren bereitstellen. Beide Klassen kryptographischer Protokolle sind nicht betroffen von Shors Algorithmus, sondern nur von den Auswirkungen durch Grovers Algorithmus.

Eine Verdoppelung der Schlüssellängen und eine Verwendung von Hashfunktionen, die längere Hashwerte liefern, z. B. AES-256 und SHA-384, bieten bereits ein angemessenes Sicherheitsniveau im Quantencomputer-Kontext.

Demnach ist es ausreichend, bei der Aushandlung der SAs für ESP und AH die Schlüssellängen der eingesetzten Kryptoverfahren entsprechend zu vergrößern und nur Authentifizierungsverfahren vorzuschlagen und anzunehmen, die entsprechend robuste Hashfunktionen verwenden. Diese SA-Aushandlung ist allerdings nicht Teil der ESP- und AH-Protokolle. Die Schutzmechanismen gegen das zugrunde gelegte Angreifermodell 2 bleiben somit unberührt.

4.2 Änderungen an IKEv2

Auch das IKE-Protokoll verwendet hashbasierte und symmetrische Kryptoverfahren, die mit doppelten Schlüssellängen und robusteren Hashfunktionen, wie z. B. AES-256 und SHA-384, einem Angriff mit Grovers Algorithmus standhalten können.

[14] Bei einem Man-in-the-Middle Angriff fängt ein Angreifer die Kommunikationsnachrichten zweier Teilnehmer ab, modifiziert die Nachrichten entsprechend seinen Wünschen und sendet sie anschließend unter der Identität des ursprünglichen Senders weiter an den Empfänger. Somit denken die Kommunikationspartner mit dem jeweils anderen legitimen Teilnehmer zu kommunizieren, obwohl sie dies tatsächlich mit dem Angreifer tun.

Zusätzlich jedoch führt IKE einen DH-Schlüsselaustausch durch, der Basis für alle kryptographischen Schlüssel einer IPsec-Sitzung ist und dessen Brechen die Sicherheit aller Schutzziele und Sicherheitsdienste von IPsec kompromittieren würde. Diese und weitere kritische Bereiche einer IKE-Sitzung müssen sowohl identifiziert als auch angepasst und der Schutz gegenüber dem zugrunde gelegten Angreifermodell 2 evaluiert werden.

IKE_SA_INIT

```
   Alice                                Bob
   ------------------------------------------------------------------
1.  HDR, SAi1, KEi, Ni   -->
2.                                      <--  HDR, SAr1, KEr, Nr, [CERTREQ]
```

Listing 4: Änderungen im IKE_SA_INIT.

Die Liste $\mathbf{SA_{i1}}$ der unterstützten kryptographischen Algorithmen und deren Parameter für die IKE SA muss die doppelten Schlüssellängen und die Verwendung von Authentifizierungsalgorithmen und PRFs mit genügend starken Hashfunktionen widerspiegeln. Wenigstens einer dieser Vorschläge muss von Bob im $\mathbf{SA_{r1}}$ angenommen werden. Das bedeutet, er muss die Algorithmen und Schlüssellängen ebenfalls unterstützen.

Der DH-Schlüsselaustausch im IKE_SA_INIT kann mithilfe von Shors Algorithmus gebrochen werden und muss somit durch ein geeignetes Verfahren zum Austausch kryptographischer Schlüssel ersetzt bzw. erweitert werden, welches einem Angriff durch Quantencomputer standhalten kann. Dies erfordert eine Änderung oder Erweiterung im Austausch des Schlüsselmaterials $\mathbf{KE_i}$ und $\mathbf{KE_r}$ und könnte je nach eingesetztem Verfahren für diesen Schlüsselaustausch und dessen Realisierung in **weiteren benötigten Nachrichten** zwischen Alice und Bob resultieren. Das wäre beispielsweise der Fall, wenn asymmetrische Schlüssel ausgetauscht werden müssen, die die Größe einer IKE-Nachricht überschreiten und somit in mehrere Nachrichten aufgeteilt werden müssen.

Ein Austausch mehrerer Nachrichten zur Realisierung des kritischen und wichtigen geheimen Schlüsselaustauschs könnte einem Angreifer aus dem Angreifermodell 1 und 2 die Möglichkeit eines Denial-of-Service Angriffs erheblich vereinfachen. Ein IPsec-Endgerät muss sich bereits empfangene Nachrichten des Schlüsselaustauschs solange merken, bis der Schlüsselaustausch vollendet wurde oder aufgrund von Timeouts abgebrochen werden muss. Sendet ein Angreifer viele Nachrichten zur Initiierung eines Schlüsselaustauschs ohne diesen Vorgang zu vollenden, werden wichtige Ressourcen des IPsec-Endgeräts belegt.

IKE_AUTH

```
   Alice                                Bob
   ------------------------------------------------------------------
3.  HDR, SK{IDi, [CERT,] [CERTREQ,]
     [IDr,] AUTH, SAi2,
     TSi, TSr}          -->
4.                                      <--  HDR, SK{IDr, [CERT,] AUTH,
                                                 SAr2, TSi, TSr}
```

Listing 5: Änderungen im IKE_AUTH.

Auch im IKE_AUTH müssen die doppelten Schlüssellängen und stärkeren Hashfunktionen in den $\mathbf{SA_{i2}}$ und $\mathbf{SA_{r2}}$ vorgesehen werden.

Die Authentifizierung der bis dahin ausgetauschten IKE-Nachrichten ist kritisch für die gesamte Senderauthentifizierung des IPsec-Protokolls und kann je nach Konfiguration den Einsatz von digitalen Zertifikaten oder von schlüsselbasierten HMACs beinhalten. Im letzten Fall müssen keine Änderungen vorgenommen werden, da die HMAC-Verfahren geheime Schlüssel verwenden, die durch die Verwendung von PQC im IKE_SA_INIT bereits abgesichert wären. Um allerdings eine personenbezogene Authentifizierung durchzuführen, ist der Einsatz von digitalen Zertifikaten nötig, mit denen die Identität der Teilnehmer nachgewiesen werden kann.

Alle aktuell eingesetzten digitalen Zertifikate, die auf der Faktorisierung oder dem Diskreten Logarithmus Problem basieren, können durch Shors Algorithmus gebrochen werden, was die Erstellung gefälschter digitaler Zertifikate und Signaturen ermöglicht.

Demnach müssen digitale Zertifikate in **CERT** eingesetzt werden, die auf PQC basieren und mit denen Signaturen für **AUTH** erstellt werden können.

CREATE_CHILD_SA

	Alice		Bob
5.	HDR, SK{**SA**, N_i, **[KE_i]**, TS_i, TS_r}	-->	
6.		<--	HDR, SK{**SA**, N_r, **[KE_r]**, TS_i, TS_r}

Listing 6: Änderungen im CREATE_CHILD_SA.

Ebenso, wie in allen anderen IKE-Schritten müssen die doppelten Schlüssellängen und stärkeren Hashfunktionen in den **SAs** von Alice und Bob vorgesehen werden.

Das Schlüsselmaterial für den optionalen erneuten DH-Schlüsselaustausch in den KE-Payloads wird zwar vertraulich und integer durch den Einsatz von $SK\{\ldots\}$ zwischen Alice und Bob ausgetauscht. Demnach kann nur ein Angreifer, der den initialen Schlüsselaustausch im IKE_SA_INIT oder die Kryptoverfahren von $SK\{\ldots\}$ brechen kann, auch an das Schlüsselmaterial KE_i und KE_r herankommen. Allerdings muss für die Erhaltung der PFS auch dieser Schlüsselaustausch durch ein Quantencomputer-resistentes Verfahren ersetzt bzw. erweitert und somit eine Änderung oder Erweiterung im Austausch des Schlüsselmaterials $\mathbf{KE_i}$ und $\mathbf{KE_r}$ durchgeführt werden.

5 Geeignete PQC-Schlüsselaustauschverfahren

Das folgende Kapitel soll einen Überblick über mögliche Verfahren für PQC sowie Herausforderungen im praktischen Einsatz liefern. Dabei wird der Fokus auf asymmetrische Verfahren

gelegt, die den DH-Schlüsselaustausch im Kontext von IKEv2 ersetzen können. Diese Analyse ist Grundlage für Kapitel 6, in dem ein ausgewähltes PQC-Verfahren für den vertraulichen Schlüsselaustausch im IKE_SA_INIT praktisch umgesetzt wird. Wie in Abschnitt 3.3 aufgezeigt, ist dieser Schlüsselaustausch kritisch für alle Schutzziele und Sicherheitsdienste von IPsec und erster Angriffspunkt einer IPsec-Sitzung durch das Angreifermodell 2.

Ein im klassischen Kontext beweisbar oder praktisch sicheres kryptographisches System gilt als sicher gegenüber Quantencomputern, sofern bekannte Quantenalgorithmen keine existentiellen Angriffe auf dieses System darstellen. Insbesondere gilt ein solches Kryptoverfahren als potentielles PQC-Verfahren, sofern bisher kein Weg gefunden wurde Shors Algorithmus darauf anzuwenden. Die Forschung im Bereich PQC hat drei Klassen von kryptographischen Systemen identifiziert, die jeweils vielversprechende PQC-Verfahren für die Ersetzung des DH-Schlüsselaustauschs beinhalten [5].

5.1 Multivariate Kryptographie

Multivariate Kryptographie stützt sich auf eine Menge von quadratischen Polynomen mit jeweils mehr als einer Variablen über einem endlichen Körper. Diese Polynommenge wird auch polynomielle Abbildung genannt. Die Sicherheit dieser kryptographischen Systeme basiert auf der Annahme, dass das sogenannte Multivariate Quadratic ($\mathcal{MQ}$) Problem – das Lösen zufälliger nichtlinearer Gleichungssysteme mit mehreren Variablen – und das Isomorphism of Polynomials Problem (IP Problem) – das Finden von invertierbaren linearen Transformationen, um zwei Mengen von quadratischen Polynomen ineinander überführen zu können – schwer zu lösen sind [21, 11]. Man spricht auch von $\mathcal{MQ}$-basierten Kryptoverfahren. Viele wichtige Vertreter, wie das MIC* Kryptosystem und HFE-Verfahren, sind in ihrer ursprünglichen Form bereits gebrochen worden [5]. Dennoch gibt es zahlreiche Weiterentwicklungen oder vorgeschlagene Änderungen basierend auf diesen Verfahren, wie z. B. die Minus(–)/Plus(+)-, die Vinegar-, die Perturbation- und die Projektion-Erweiterung [5]. Diese haben zwar erst sehr wenig Vertrauen in kryptographischen Kreisen erlangt, dennoch bieten sie vielversprechende Ansätze. Aktuell als noch ungebrochen gilt das Verfahren Perturbed Matsumotu-Imai Plus (PMI+) [10] als Kandidat für asymmetrische Verschlüsselung, mit dem Vorteil der sehr effizienten Ver- und Entschlüsselung. Allerdings sind die Forschung an $\mathcal{MQ}$-basierten Kryptosystemen und die zugrunde liegende Mathematik noch sehr jung, wodurch das geringe Vertrauen in deren Sicherheit begründet ist.

5.2 Gitterbasierte Kryptographie

Auf Grundlage von verschiedenen Gitterproblemen, z. B. der Reduktion der Basisvektoren eines Gitters auf möglichst kurze Vektoren, die ebenfalls als Basis für das selbe Gitter dienen können, wurden Trapdoor-Funktionen für asymmetrische Verschlüsselungssysteme entwickelt. Mithilfe der Arbeit von [1] konnte die Schwierigkeit dieser Gitterprobleme gezeigt und später sogar in speziellen Instanzen als $\mathcal{NP}$-hart bewiesen werden. Das ermöglichte sehr starke Sicherheitsbeweise für gitterbasierte Kryptoverfahren, die auf diesen Gitterprobleminstanzen basieren [5]. Diese beweisbar sicheren Verfahren weisen allerdings einen Mangel an Effizienz

oder einen zu hohen Speicherverbrauch oder beides auf, um praktische Relevanz zu erlangen. Für effizientere gitterbasierte Verfahren, die großes Potential für den praktischen Einsatz bergen, fehlen wiederum die starken Sicherheitsbeweise, was das Vertrauen in ihre Sicherheit schwächt. Wichtigster Vertreter der gitterbasierten Verschlüsselungssysteme ist das Number Theory Research Unit (NTRU) Verfahren [13], welches eine hohe Effizienz und niedrigen Speicherverbrauch bietet. Allerdings existiert kein Sicherheitsbeweis und es unterliegt zudem patentrechtlichen Beschränkungen.

5.3 Codebasierte Kryptographie

Fehlerkorrigierende Codes wurden ursprünglich eingeführt und untersucht, um Übertragungsfehler von Informationen über störanfällige Kanäle korrigieren zu können. Dabei werden Nachrichten mithilfe spezieller Codes kodiert, die besondere Eigenschaften haben. Verfälschungen der Nachrichten durch beispielsweise technisch bedingtes Rauschen eines Übertragungskanals können somit erkannt und durch entsprechende Dekodierung entfernt werden. Dabei werden die Kenntnis des speziellen Codes und dessen Eigenschaften ausgenutzt. Im Jahr 1978 wurde das Dekodierungsproblem für zufällige lineare Codes, bei dem versucht wird die Fehler in einer kodierten Nachricht ohne Kenntnis des zugrundeliegenden Codes zu entfernen, als $\mathcal{NP}$-vollständig bewiesen [4]. Erstmals genutzt wurde das Dekodierungsproblem als Trapdoor-Funktion im McEliece-Verfahren [22], welches etwa zum gleichen Zeitpunkt wie das RSA-Verfahren publiziert und gegen das bis heute kein existenzieller Angriff gefunden wurde. Trotz seiner starken Sicherheitsbeziehungen und hohen Effizienz durch den niedrigen Rechenaufwand bei der Ver- und Entschlüsselung, konnte es keine praktische Relevanz erreichen. Dies resultiert hauptsächlich aus den im Vergleich zu allen andern PQC-Kandidaten sehr großen öffentlichen Schlüsseln. Ein weiterer wichtiger Vertreter der codebasierten Kryptosysteme ist das auf dem McEliece-Verfahren basierende Niederreiter-Verschlüsselungssystem [25], welches im Kontext eines asymmetrischen Schlüsselaustauschs von allen PQC-Kandidaten die meisten Vorteile für einen produktiven Einsatz in IPsec bietet. Die Sicherheit des Niederreiter-Verfahrens ist äquivalent zur Sicherheit des McEliece-Verfahrens, wie in [20] gezeigt werden konnte. Ausschlaggebend für die Verwendung des Niederreiter-Verfahrens in der vorliegenden Arbeit ist die im Kontext der Verschlüsselung von zufälligen Nachrichten als kryptographische Schlüssel bessere Performanz gegenüber dem McEliece-Verfahren [9]. Zudem bietet es eine höhere praktische Sicherheit gegenüber dem PMI+ Verfahren und eine sehr hohe Ver- und Entschlüsselungsperformanz im Vergleich mit aktuellen asymmetrischen Verfahren, wie RSA und sogar mit dem NTRU-Verfahren [7].

6 PQC Prototyp für IPsec

Um negative Seiteneffekte und Abschwächungen der IPsec- und IKE-Sicherheitsdienste zu vermeiden, wurde mit der Vorgabe gearbeitet, den Ablauf des IKE-Protokolls – also den Austausch zweier Nachrichten für den Schlüsselaustausch im IKE_SA_INIT-Schritt – nicht zu verändern und keine neuen potentiellen Angriffspunkte einzuführen. Der neue Niederreiter-Schlüsselaustausch sieht demnach folgendermaßen aus:

Bit-Sicherheit	Goppa Code Parameter (n,t,m)
80	$(1844,56,11)$
128	$(3216,70,12)$
256	$(6974,138,13)$

Tabelle 3: Angepasste Code Parameter der binären Goppa Codes. In Anlehnung an [6].

- Der Initiator generiert ein Niederreiter-Schlüsselpaar und sendet seinen öffentlichen Schlüssel in der ersten IKE_SA_INIT-Nachricht an den Empfänger.
- Der Empfänger generiert ein zufälliges gemeinsames Geheimnis, verschlüsselt dieses mit dem öffentlichen Schlüssel des Initiators und sendet das verschlüsselte gemeinsame Geheimnis in der zweiten IKE_SA_INIT-Nachricht an den Initiator zurück.

6.1 Praktische Umsetzung

Bei der Implementierung des Niederreiter-Schlüsselaustauschs wurden die in [6] konkret vorgeschlagenen Parameter der zugrunde liegenden binären Goppa Codes geringfügig verändert, um eine einfachere Berechnung der Niederreiter-Kontrollmatrizen zu ermöglichen. Die Änderungen bieten nach der von den Autoren publizierten Formel zur Berechnung des binären Arbeitsfaktors

$$min\left\{\frac{1}{2}\cdot\binom{n}{t}\cdot\binom{n-k}{t-x}^{-1}\cdot\binom{k}{x}^{-1/2} : x \geq 0\right\},$$

mit n, t und $k = n - m \cdot t$ als Codeparameter, dennoch eine äquivalente Bit-Sicherheit. Tabelle 4 zeigt die neuen Parameter. Bei der Verschlüsselung werden zufällig erzeugte Vektoren als Nachrichten mit der öffentlichen Kontrollmatrix des Kommunikationspartners zu einem sogenannten Syndrom multipliziert. Die Entschlüsselung verwendet die private Kontrollmatrix und einige Goppa Code spezifische Informationen sowie den vorhandenen Dekodierungsalgorithmus, um aus dem Syndrom den Vektor zu ermitteln, der gleichzeitig die entschlüsselte Nachricht darstellt.

Weiterhin wurde auf das Open-Source Projekt *strongSwan* in der Version 5.0.1 zurück gegriffen, mit dessen Hilfe ein IPsec-basiertes Virtuelles Privates Netzwerk (VPN) erstellt werden kann.[15] Der Kern der Software ist die Implementierung des IKEv2-Protokolls, die für die Erzeugung und Aushandlung von IKE und Child SAs verwendet werden kann. Das Projekt ist modular aufgebaut und kann durch unterschiedliche Plugins in seiner Funktionalität erweitert werden. Eine dieser Funktionalitäten ist der DH-Schlüsselaustausch, der im IKE_SA_INIT-Schritt durchgeführt wird und mit dem neuen Niederreiter-Schlüsselaustausch ersetzt werden sollte. Dafür wurde ein neues Plugin geschrieben, welches anstelle eines DH-Plugins verwendet werden kann und die Funktionalitäten des Niederreiter-Verfahrens zum Zwecke eines Schlüsselaustauschs gebraucht. Durch die Implementierung festgelegter Schnittstellen für DH-Plugins kann strongSwan somit das neue Plugin transparent als DH-Plugin verwenden und muss nicht weiter an den neuen Schlüsselaustausch angepasst werden.

[15] Entwickelt und unterhalten wird strongSwan von der University of Applied Sciences in Rapperswil, Schweiz. Nähere Informationen findet man unter `http://www.strongswan.org/`.

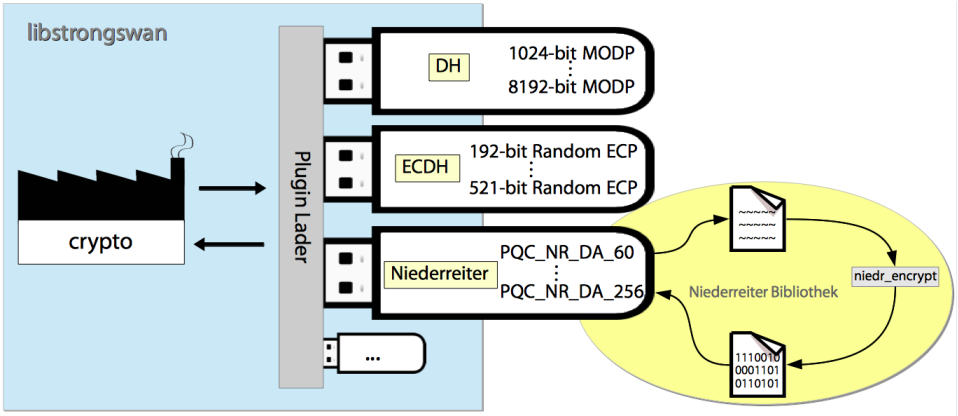

Abbildung 1: Integration des Niederreiter-Schlüsselaustauschs in strongSwan als neues Plugin.

6.2 Schlüsselaustauschplugin

In Listing 7 findet man die wichtigsten vom Schlüsselaustauschplugin verwendeten Schnittstellen der Niederreiter-Implementierung und das Strukturobjekt einer Niederreiter-Instanz, welches alle relevanten Informationen, wie z. B. Codeparameter und den öffentlichen bzw. privaten Schlüssel, enthält.

```
1  typedef struct niedr_t {
2     // dimensions and max. correctable errors of the underlying Goppa code
3     niedr_params_t params;
4     // public parity check matrix and max. correctable errors
5     niedr_pubkey_t *pubkey;
6     // Goppa code specific information for decoding
7     niedr_privkey_t *privkey;
8  }
9
10 niedr_t *niedr_new(int niedr_group);
11 void niedr_keygen(niedr_t *nr);
12 syndrome_t niedr_encrypt(binvector_t errvect, niedr_pubkey_t *pubkey);
13 binvector_t niedr_decrypt(syndrome_t *syndrome, niedr_privkey_t *privkey);
14 binvector_t niedr_random_binvector(niedr_pubkey_t *key);
```

Listing 7: Das Niederreiter-Strukturobjekt und die Funktionsprototypen der Niederreiter-Bibliothek.

Das Niederreiter-Plugin implementiert die gleichen Schnittstellen, wie ein normales DH-Plugin. Es registriert sich beim Start von strongSwan am Kryptofabrikmodul von charon mit der Information, welche speziellen DH-Algorithmen es implementiert, die eindeutig sind sowie eine eigens zugewiesene DH-Gruppennummer aus dem privaten Bereich zwischen 1024 und 65535 besitzen.[16] Bei Konfiguration dieser speziellen DH-Algorithmen im strongSwan wird somit das Niederreiter-Plugin zur Durchführung des initialen Schlüsselaustauschs geladen und verwendet. Abbildung 1 veranschaulicht dieses Funktionsprinzip. Nachfolgend werden die implementierten Schnittstellen näher erläutert.

[16] Der für den privaten Einsatz vorgesehene DH-Gruppenbereich wird von der Internet Assigned Numbers Authority (IANA) in [15] definiert.

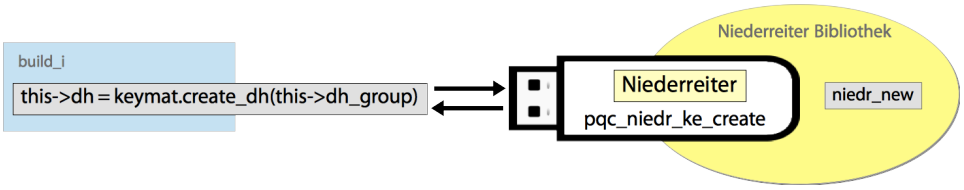

Abbildung 2: Initialisierung des Niederreiter-Plugins durch das Kryptofabrikmodul.

pqc_niedr_ke_create

Diese Methode wird vom Kryptofabrikmodul zur Erzeugung einer Schlüsselaustausch-Instanz des Plugins aufgerufen. Sie initialisiert alle öffentlichen und privaten Attribute in der Plugin-eigenen Struktur pqc_niedr_ke_t und zusätzlich – im Gegensatz zu anderen DH-Plugins – eine der angeforderten Bit-Sicherheit entsprechende Niederreiter-Instanz.

```
1  typedef struct pqc_niedr_ke_t {
2     niedr_t nr;
3     syndrom_t syndrome;
4     chunk_t shared_secret;
5     bool initiator;
6  }
7
8  pqc_niedr_ke_t *pqc_niedr_ke_create(diffie_hellmann_group_t group) {
9     pqc_niedr_ke_t *this;
10
11    this = init_pqc_niedr_plugin(group);
12
13    switch(group) {
14       case PQC_NR_DA_60:
15          this->nr = niedr_new(0); break;
16       case PQC_NR_DA_80:
17          this->nr = niedr_new(1); break;
18                ⋮
19       case PQC_NR_DA_256:
20          this->nr = niedr_new(10); break;
21    }
22
23    return this;
24 }
```

Listing 8: Code der pqc_niedr_ke_create Methode, die zur Initialisierung des Niederreiter-Plugins aufgerufen wird.

destroy

Die destroy Methode wird aufgerufen, sobald die Plugin-Instanz nicht mehr gebraucht wird. Das ist entweder der Fall, wenn der Schlüsselaustausch im IKE_SA_INIT Schritt abgeschlossen wurde oder in einem Fehlerfall.

Sie ermöglicht die Freigabe von alloziertem Speicher und das Löschen von sensiblen Informationen über den vertraulichen Schlüsselaustausch.

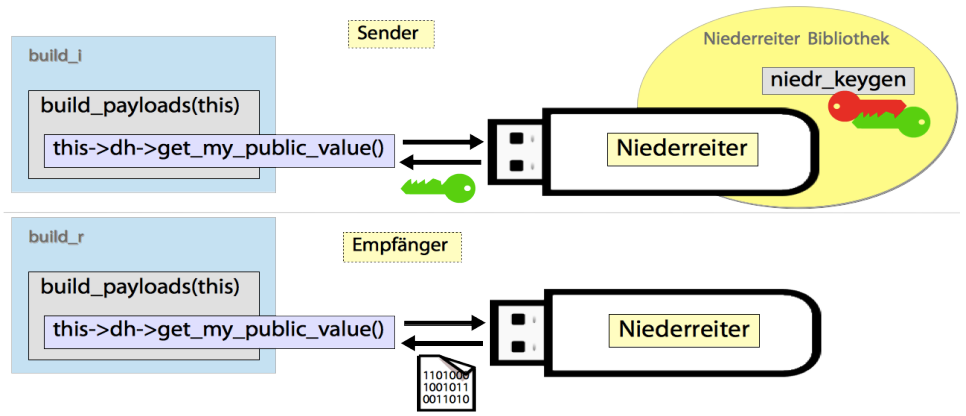

Abbildung 3: Aufruf der Methode get_my_public_value im Niederreiter-Plugin jeweils durch den Sender und Empfänger.

get_dh_group

Diese Methode liefert die bei Erstellung der Plugin-Instanz übergebene DH-Gruppe zurück. Sie wird bei der Erzeugung der KE_i und KE_r-Payloads aufgerufen, um den jeweils verwendeten DH-Algorithmus auf Sender- und Empfängerseite aufzuzeigen.

get_shared_secret

Nach dem erfolgreichen Schlüsselaustausch wird vom keymat Objekt durch Aufruf dieser Methode das gemeinsame Geheimnis erfragt, um damit alle weiteren kryptographischen Schlüssel für die IKE SA und potentiell auch für daraus ableitbare Child SAs zu erzeugen.[17]

get_my_public_value

Die Implementierungen der bisherigen Methoden beinhalten keine großen Unterschiede zu anderen DH-Plugins. Doch in der *get_my_public_value* Methode muss zwischen Sender und Empfänger unterschieden werden, wie in Abbildung 3 veranschaulicht.

Der Sender generiert ein neues Niederreiter-Schlüsselpaar und liefert in einer von charon vorgegebenen *chunk_t* Datenstruktur den eigenen öffentlichen Schlüssel zurück, der daraufhin als KE_i-Payload in der ersten IKE-Nachricht den Empfänger erreicht.

Der öffentliche Wert des Empfängers ist ein als Syndrom verschlüsseltes gemeinsames Geheimnis, welches mit dieser Methode abgerufen und als KE_r-Payload in der zweiten IKE-Nachricht an den Sender gesendet wird. Um das Syndrom erstellen zu können, muss der Empfänger zuvor

[17] Für jede Child SA kann ein neuer DH-Schlüsselaustausch durchgeführt werden, aus dem sich neue kryptographische Schlüssel ergeben. Es müssen somit nicht die hier angesprochenen Schlüssel aus der IKE SA verwendet werden.

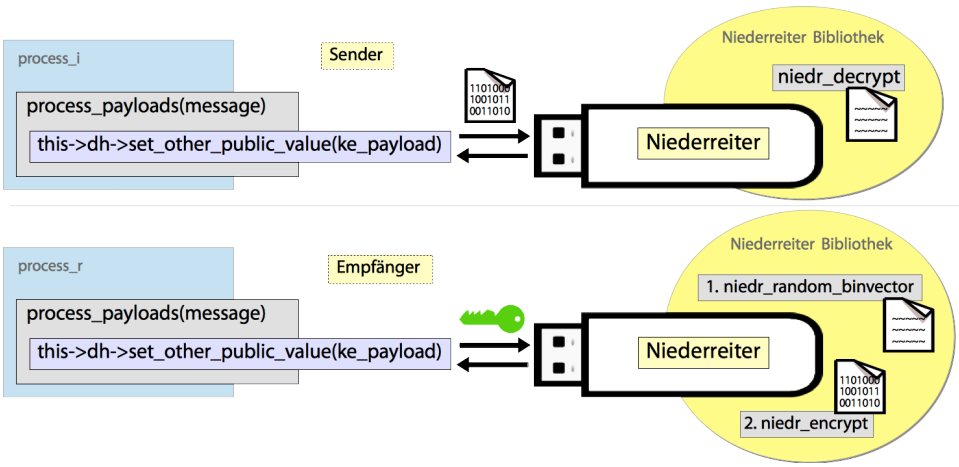

Abbildung 4: Aufruf der Methode set_other_public_value im Niederreiter-Plugin jeweils durch den Sender und Empfänger.

den öffentlichen Schlüssel des Senders durch Aufruf der Methode *set_other_public_value* erhalten haben.

```
1 void get_my_public_value(pqc_niedr_ke_t *this, chunk_t *value) {
2    if (NULL == this->syndrome) { /* Initiator */
3       this->initiator = true;
4       niedr_keygen(this->nr);
5       *value = pubkey2chunk(this->nr->pubkey);
6    } else { /* Responder */
7       *value = syndrome2chunk(this->syndrome);
8    }
9 }
```

Listing 9: Code der get_my_public_value Methode, die zur Abfrage des öffentlichen Schlüsselmaterials für den Inhalt des KE-Payloads aufgerufen wird.

set_other_public_value

Auch in dieser Methode muss zwischen den beiden Kommunikationspartnern unterschieden werden. Der Sender erhält das Syndrom des Empfängers als Inhalt des KE_r-Payloads der zweiten IKE-Nachricht. Dieses entschlüsselt er mithilfe des privaten Niederreiter-Schlüssels und erhält somit das vom Empfänger generierte gemeinsame Geheimnis.

Wie bereits erwähnt, erhält der Empfänger mit dieser Methode den öffentlichen Niederreiter-Schlüssel des Senders. Mit dessen Hilfe generiert er einen zufälligen binären Vektor, der ein Hamminggewicht entsprechend den Parametern des öffentlichen Schlüssels besitzt, verschlüsselt ihn und speichert den generierten Vektor als gemeinsames Geheimnis. Das Ergebnis der Verschlüsselung ist der öffentliche Wert des Empfängers und wird als Syndrom für dessen Abfrage durch die get_my_public_value Mathode gespeichert.

```
1 void set_other_public_value(pqc_niedr_ke_t *this, chunk_t value) {
2    binvector_t errvect;
```

```
3
4       if (this->initiator) { /* Initiator */
5           syndrome_t syndrome = chunk2syndrome(&value);
6           errvect = niedr_decrypt(&syndrome, this->nr->privkey);
7       } else { /* Responder */
8           this->nr->pubkey = chunk2pubkey(&value);
9           errvect = niedr_random_binvector(this->nr->pubkey);
10          this->syndrome = niedr_encrypt(&errvect, this->nr->pubkey);
11      }
12
13      this->shared_secret = binvector2chunk(errvect);
14  }
```

Listing 10: Code der set_other_public_value Methode, die zur Speicherung des öffentlichen Schlüsselmaterials des Kommunikationspartners aufgerufen wird.

Eine Veranschaulichung der Methode mit Unterscheidung zwischen Sender und Empfänger findet sich in Abbildung 4.

7 Probleme und Lösungen

Die größte Herausforderung bei der Implementierung des Niederreiter-Schlüsselaustauschs war die Verteilung der öffentlichen Schlüssel. Daraus ergab sich das Problem der *IKE-Nachrichtenfragmentierung*, dessen Umgehung weitere Schwierigkeiten nach sich zog. Diese Probleme und deren Lösungen werden nachfolgend erläutert.

7.1 Schlüsselgrößen

Das Niederreiter-Plugin in seiner ursprünglichen Version kann problemlos verwendet werden, solange die Größe des öffentlichen Schlüssels im KE_i-Payload der ersten IKE-Nachricht einen gewissen Wert nicht überschreitet. Dieser Wert ist abhängig von der Implementierung des IKE-Standards, in dem diesbezüglich Folgendes zu finden ist [16]:

> „All IKEv2 implementations MUST be able to send, receive, and process IKE messages that are up to 1280 octets long, and they SHOULD be able to send, receive, and process messages that are up to 3000 octets long. IKEv2 implementations need to be aware of the maximum UDP message size supported [...].“

Die Grenze von 3000 Bytes entspricht der Größe einer IKE-Nachricht, in der die Kontrollmatrix eines binären $(n = 115, t = 12, m = 7)$ Goppa Codes in der vorliegenden Implementierung versandt wird. Dieser Code bietet nach der Einschätzung von [6] eine Sicherheit von 5 Bit. Sollen größere Codes zur Erhöhung der Sicherheit verwenden werden, kommt es darauf an, ob die IKE-Implementierung entgegen des Standards größere Nachrichten verarbeiten kann.

Bit-Sicherheit	DH	Niederreiter
80	128	167.440
128	384	443.088
200	1.024	1.097.560

Tabelle 4: Datenmenge in Byte für einen Schlüsselaustausch vom Sender zum Empfänger.

Wie in Tabelle 4 zu sehen, handelt es sich bei aktuell empfohlener Bit-Sicherheit des Niederreiter-Verfahrens um Kontrollmatrizen in den Größenordnungen von mehreren hundert Kilobyte. Das übersteigt nicht nur die im IKE-Standard vorgeschlagenen 3000 Bytes, sondern auch die maximale User Datagram Protocol (UDP) Nachrichtengröße von 65535 Bytes.[18] Eine Fragmentierung der Nachrichtenpakete ist somit unausweichlich.

Das damit zusammenhängende Problem ist die Blockierung fragmentierter UDP-Pakete durch viele Router und Firewalls zur Unterbindung spezielle Angriffe [33]. Als Lösung gibt es die folgenden zwei Ansätze.

7.1.1 IKE-Fragmentierung

Mithilfe der vorliegenden Informationen über Header- und Payloadgrößen der IKE-Nachrichten können diese vor dem Verpacken in UDP-Pakete bereits in mehrere kleine Nachrichten aufgeteilt werden. Das verschiebt zwar die Aufgaben der korrekten und robusten (De-)Fragmentierung in die IKE-Implementierung. Allerdings werden somit nur unfragmentierte UDP-Pakete versandt, die von den erwähnten Netzwerkfiltern nicht blockiert werden.

Für IKEv2 ist dieser Ansatz noch nicht standardisiert und wird aktuell auch nicht von strongSwan, sondern nur von proprietären Implementierungen umgesetzt.[19]

7.1.2 Hash und URL

Für die Umgehung der Nachrichtenfragmentierung bietet der IKE-Standard den folgenden als '*Hash und URL*' bezeichneten Lösungsansatz an [16]:

> „Hash and URL encodings allow IKE messages to remain short by replacing long data structures with a 20-octet SHA-1 hash [...] of the replaced value followed by a variable-length URL that resolves to the [...] data structure itself."

[18] IKE-Nachrichten werden als UDP-Pakete versandt. Diese sehen ein Headerfeld für die Größe des UDP-Pakets von 16 Bit vor, was in einer maximalen Paketgröße von 65535 Bytes inklusive Header resultiert. Eine Fragmentierung von UDP Paketen passiert allerdings schon bei der Überschreitung der Maximum Transmission Unit (MTU), die in IP-Netzwerken häufig bei 1500 Bytes liegt.

[19] Cisco unterstützt ab dem Release 12.4(15)T7 der *Cisco IOS Software* die Fragmentierung von IKE-Nachrichten. Siehe `http://www.cisco.com/en/US/docs/ios-xml/ios/sec_conn_ikevpn/configuration/15-2mt/sec-fragment-ike-pack.html`.

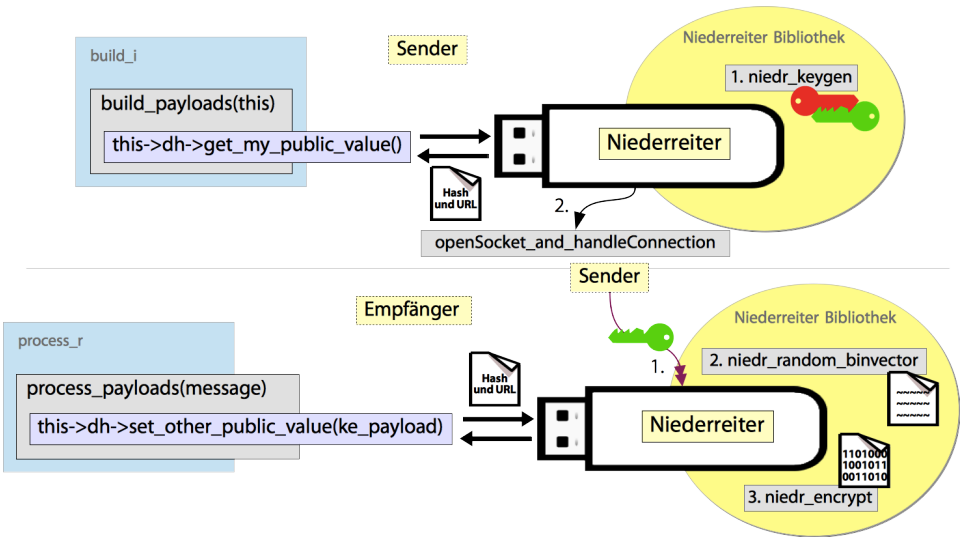

Abbildung 5: Aufruf der Methode get_my_public_value auf Senderseite und der Methode set_other_public_value auf Empfängerseite beim Einsatz von ‘Hash und URL’ und eigenem Socket im Niederreiter-Plugin.

Dieser Ansatz bietet für die Lösung des vorliegenden Problems im Niederreiter-Plugin einen entscheidenden Vorteil. Die IKE-Implementierung muss nicht geändert werden, da die Verwendung von ‘Hash und URL’ Payloads transparent umgesetzt werden kann. Darüber hinaus existieren bereits Plugins zum Herunterladen von entfernten Ressourcen, wie z.B. das *cURL*-Plugin. Aus diesem Grund wurde der Niederreiter-Schlüsselaustausch im IKE_SA_INIT Schritt für die Verwendung von ‘Hash und URL’ angepasst.

7.2 Funktionierende Lösung

Beim Aufruf der Methode *get_my_public_value* liefert der Sender nicht mehr den öffentlichen Schlüssel zurück, sondern erstellt einen SHA1 Hash dieses Schlüssels, erstellt eine URL, die auf das IPsec-Endgerät des Senders selbst verweist und öffnet einen Socket in einem separatem Thread, mit dem sich der Empfänger nach Erhalt des ‘Hash und URL’ Payloads verbinden kann. Über diesen Socket wird der Schlüssel des Senders an den Empfänger übertragen.[20] Der auf den öffentlichen Schlüssel verweisende ‘Hash und URL’ String wird vom Sender wie zuvor in einem chunk_t Objekt zurückgeliefert, welcher als KE_i-Payload der ersten IKE-Nachricht versandt wird.

```
1 void get_my_public_value(pqc_niedr_ke_t *this, chunk_t *value) {
2    if (NULL == this->syndrome) { /* Initiator */
3       this->initiator = true;
4       niedr_keygen(this->nr);
5
```

[20] Prinzipiell kann jeder von den IPsec Endgeräten erreichbare Server als Netzwerkknoten dienen, auf dem der öffentliche Schlüssel des Senders abgelegt wird. Er dient nur zum Austausch des öffentlichen Schlüsselmaterials und führt keine neuen Möglichkeiten ein, dieses Schlüsselmaterial unbemerkt und erfolgreich zu ändern. Die Variante mit der URL auf sich selbst führt allerdings keinen neuen Single Point of Failure (SPOF) in das System ein und erspart das Hochladen des Schlüssels.

```
6         chunk_t hash = getHash(this->nr->pubkey);
7         chunk_t url = createURL_to_self();
8
9         // Concatenates two chunk_t structures
10        *value = chunk_cat("cc", hash, url);
11
12        // Creates a job with priority JOB_PRIO_CRITICAL
13        job_t *socketJob = (job_t *)callback_job_create_with_prio(
14           openSocket,
15           this->nr->pubkey,
16           NULL, NULL,
17           JOB_PRIO_CRITICAL);
18
19        // Lets a worker process the job in a separate thread
20        lib->processor->queue_job(lib->processor, socketJob);
21     } else { /* Responder */
22              ⋮
23     }
24  }
```

Listing 11: Code der get_my_public_value Methode, die den 'Hash und URL' Payload Ansatz mit eigenem Socket umsetzt. Für die Empfängerseite muss nichts geändert werden.

Der Empfänger erhält den Hash und die URL mit der Methode *set_other_public_value*, lädt sich den öffentlichen Schlüssel des Senders mithilfe der URL herunter und prüft den errechneten Hash des besorgten Schlüssels mit dem vom Sender mitgesendeten Hash. Stimmen beide überein, kann der Empfänger mit der Erzeugung des gemeinsamen Geheimnisses und dessen Verschlüsselung fortfahren.

```
1  void set_other_public_value(pqc_niedr_ke_t *this, chunk_t value) {
2     binvector_t errvect;
3
4     if (this->initiator) { /* Initiator */
5              ⋮
6     } else { /* Responder */
7        char *url = extract_url(value);
8        char *hash = extract_hash(value);
9
10       chunk_t fetchedKey = fetchKeyFromURL(url);
11
12       if (!hashesMatch(hash, getHash(fetchedKey))) {
13          return;
14       } else {
15          this->nr->pubkey = chunk2pubkey(&fetchedKey);
16          errvect = niedr_random_binvector(this->nr->pubkey);
17          this->syndrome = niedr_encrypt(&errvect, this->nr->pubkey);
18       }
19    }
20
21    this->shared_secret = binvector2chunk(errvect);
22 }
```

Listing 12: Code der set_other_public_value Methode, die den 'Hash und URL' Payload Ansatz umsetzt. Für die Senderseite muss nichts geändert werden.

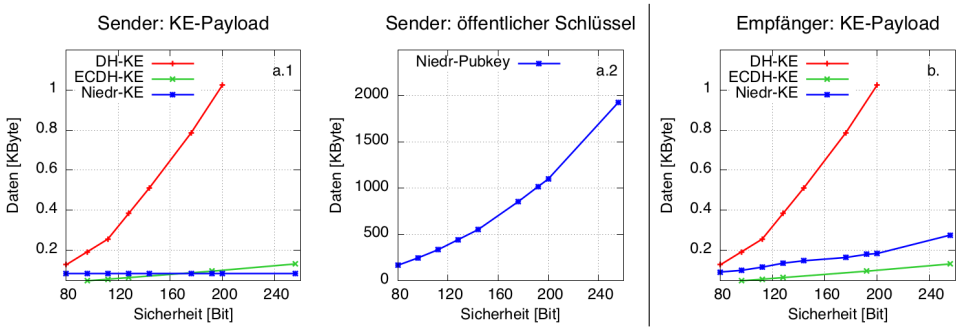

Abbildung 6: Vergleich der vom Sender (a.1 und a.2) und Empfänger (b.) übertragenen Daten für den DH-, ECDH- und Niederreiter-Schlüsselaustausch bei steigender Bit-Sicherheit. Der Sender muss mit dem Niederreiter-Plugin zunächst den KE-Payload (a.1) und darauffolgend den öffentlichen Schlüssel (a.2) senden.

Die Sockets des Senders und Empfängers, die für die Übertragung des öffentlichen Niederreiter-Schlüssels verwendet werden, mussten zusätzlich mit einer IPsec-Umgehung konfiguriert werden, sodass diese Verbindung reine IP-Pakete austauschen kann.

8 Simulationen und Evaluierung

Mit der Durchführung und Auswertung von Simulationen wurde die Performanz des Niederreiter-Plugins während des IKE_SA_INIT-Schrittes im Vergleich zur Performanz eines DH- und ECDH Schlüsselaustauschs bei vergleichbarer Bit-Sicherheit analysiert. Zwischen zwei IPsec-Endgeräten wurden dafür nacheinander mehrere IPsec-Verbindungen mit fest vorkonfiguriertem Schlüsselaustauschalgorithmus für die IKE SA Aushandlung aufgebaut. Aus den erfassten Daten wurde anschließend ein Mittelwert errechnet.

8.1 Ergebnisse

Die erhobenen Daten ermöglichten den Vergleich der Datenmengen, die für den Schlüsselaustausch zwischen zwei IPsec-Endgeräten ausgetauscht werden müssen. Abbildung 6 zeigt, dass die beiden DH-Plugins mit Datenmengen unter 1024 Bytes auskommen, die vom Initiator zum Empfänger und ebenso in die umgekehrten Richtung versendet werden müssen. Das Niederreiter-Plugin hingegen benötigt drei bis vier Größenordnungen mehr Daten, die hauptsächlich den großen öffentlichen Schlüssel beinhalten.

Darüber hinaus ermöglichten die Simulationen zum einen den Vergleich der benötigten Zeit für die kryptographischen Berechnungen während des Schlüsselaustauschs und zum anderen für die Gesamtzeit der Aushandlung einer IKE SA. Auffällig für das Niederreiter-Plugin ist die nur sehr leicht ansteigende Latenz auf Initiatorseite bei steigender Bit-Sicherheit und die annähernd konstante Latenz auf Empfängerseite, die jeweils signifikant über denen des ECDH-Plugins liegen. Im Gegensatz dazu zeigt die benötigte Zeit für die Fertigstellung einer IKE SA mithilfe des DH-Plugins bei steigender Bit-Sicherheit allerdings ein annähernd exponentielles

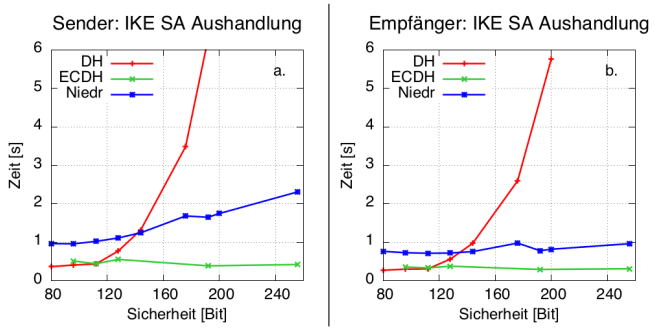

Abbildung 7: Vergleich der Zeiten für eine IKE SA Aushandlung zwischen den drei Plugins Niederreiter, DH und ECDH. Getrennte Betrachtung jeweils für Sender und Empfänger.

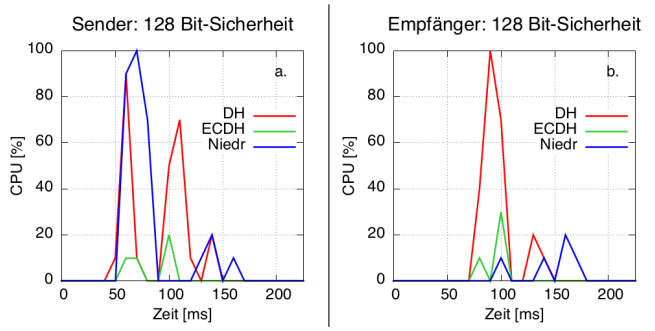

Abbildung 8: Vergleich der CPU Auslastung zwischen den drei Plugins Niederreiter, DH und ECDH bei einem Schlüsselaustausch mit 128 Bit. Getrennte Betrachtung jeweils für Sender und Empfänger.

Wachstum. Dadurch ergibt sich eine höhere und stark ansteigende Latenz im Vergleich zum Niederreiter-Plugin ab einer Sicherheit von 144 Bit, wie Abbildung 7 veranschaulicht.

Die Simulationen erlaubten des Weiteren eine vergleichende Betrachtung der benötigten Rechenleistung, die die aufgrund der Latenzen gezogenen Schlüsse über die Performanz des Niederreiter-Plugins bestätigen. Trotz des Nachteils im Vergleich zum ECDH-Plugin erzeugt das Niederreiter-Plugin nur bei der Schlüsselgenerierung auf Initiatorseite eine signifikant höhere CPU Auslastung und schneidet im Vergleich mit dem DH-Plugin insgesamt besser ab. Die Benötigte Rechenleistung einer 128-Bit sicheren IKE-SA-Aushandlung ist in Abbildung 8 zu sehen.

8.2 Vor- und Nachteile

Der größte Nachteil des Niederreiter-Schlüsselaustauschs sind die bereits erwähnten großen öffentlichen Schlüssel, die zu einem Performanzverlust gegenüber anderen Verfahren führen. Sie verursachen nicht nur hohe Latenzen bei einer IKE SA Aushandlung im Vergleich zum ECDH-Plugin, wie die Ergebnisse der Simulationen zeigen, sondern sind auch der Grund für die aufwendige Anpassung des Schlüsselaustauschs sowie für notwendige Lösungsansätze durch das Problem der IKE-Nachrichtenfragmentierung. Verstärkt werden diese Probleme durch die mangelnde Präsenz codebasierter Verfahren im praktischen Einsatz und in den

Standards kryptographischer Protokolle. Einerseits fehlen einheitliche Validierungs- und Testmöglichkeiten, die somit aufwendig erstellt werden mussten. Andererseits mussten individuelle Lösungen gefunden werden, um unerwünschte Seiteneffekten und neuen Angriffsmöglichkeiten zu umgehen. Zusätzlich stellt der neue Schlüsselaustausch eine Abweichungen vom IKE-Standard dar, was eine Nichtkompatibilität mit anderen IKE-Implementierungen nach sich zieht.

Die Simulationen zeigen allerdings auch die Vorzüge des Niederreiter-Verfahrens. Die notwendigen kryptographischen Operationen sind sehr effizient realisierbar und benötigen deutlich weniger Rechenzeit als die der DH- und ECDH-Verfahren. Daraus ergibt sich trotz der vorhandenen Nachteile eine insgesamt bessere praktische Leistung im Vergleich zum DH-Plugin bei hohen Bit-Sicherheiten. Zieht man zudem noch in Betracht, dass eine IKE SA Aushandlung nur zu Beginn einer IPsec-Sitzung und anschließend optional für den Erhalt der PFS erst nach einer vordefinierten Zeit oder ausgetauschten Datenmenge wiederholt durchgeführt wird [16], relativiert sich der allgemeine Performanznachteil des Niederreiter-Plugins gegenüber dem ECDH-Plugin.

Der Einsatz des Niederreiter-Verfahrens in IPsec bietet zusätzlich den entscheidenden Vorteil der Sicherheit gegen Quantencomputer. Ungeachtet der Tatsache, dass die praktische Realisierung universeller Quantencomputer aktuell noch auf sich warten lässt, können ab dem Zeitpunkt der praktischen Realisierung alle mit nicht-PQC-Verfahren verschlüsselten Nachrichten, auch die in der Vergangenheit gespeicherten, gebrochen und gelesen werden. Die nachhaltige Vertraulichkeit solcher verschlüsselter Nachrichten ist somit bereits heute gefährdet, kann allerdings durch die Verwendung des Niederreiter-Schlüsselaustauschs wieder hergestellt werden.

9 Ausblick

Die prototypische Implementierung des Niederreiter-Verfahrens und des strongSwan-Plugins bietet unter anderem in den folgenden Ansätzen Potential für Verbesserungen und Erweiterungen. Neben der Weiterentwicklung zu einer Projektversion, können Designentscheidungen über die Art der Einbindung des Verfahrens in strongSwan und die Umgehung der IKE-Nachrichtenfragmentierung hinterfragt und umfassend analysiert werden.

Darüber hinaus bietet die kontinuierliche Forschung auf dem Gebiet der codebasierten Kryptoverfahren immer wieder neue Ansätze und Möglichkeiten, mit den Herausforderungen der großen öffentlichen Schlüssel umzugehen [2, 24, 26]. Diese Entwicklungen sollten verfolgt und könnten zur Performanzverbesserung umgesetzt werden.

Viele der entstandenen Probleme bei der Implementierung des Niederreiter-Plugins könnten auf leichtere Art und Weise behoben werden, wenn der IKE Standard den Einsatz von PQC-Verfahren als Ersatz für den DH-Schlüsselaustausch bereits vorsehen würde und somit den Umgang mit ihren Herausforderungen spezifizieren könnte. Mit dem Wissen aus der vorliegenden Arbeit über diese Herausforderungen und Lösungen könnte eine Anpassung oder Erweiterung des Standards um die IKE-Nachrichtenfragmentierung und PQC-Schlüsselaustauschmethoden auf den Weg gebracht werden.

Abschließend kann die Implementierung von PQC in IKE und damit in IPsec vervollständigt werden, indem auch der IKE_AUTH Schritt entsprechend der theoretischen Analyse der vorliegenden Arbeit angepasst wird.

10 Danksagung

Für wertvolle Diskussionsbeiträge, inhaltliches Feedback und Textkorrekturen sei Stefan Köpsell, Thomas Egerer und Judith Zimmer gedankt.

Die dieser Arbeit zugrunde liegende Diplomarbeit wurde von der Firma secunet Security Networks AG unterstützt.

Literatur

[1] AJTAI, M. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (New York, NY, USA, 1996), STOC '96, ACM, pp. 99–108.

[2] BARBIER, M., AND BARRETO, P. S. L. M. Key reduction of McEliece's cryptosystem using list decoding. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on* (2011), pp. 2681–2685.

[3] BELLARE, M., AND ROGAWAY, P. Introduction to Modern Cryptography. In *UCSD CSE 207 Course Notes* (sep 2005), p. 207.

[4] BERLEKAMP, E., MCELIECE, R., AND VAN TILBORG, H. C. A. On the inherent intractability of certain coding problems (Corresp.). *Information Theory, IEEE Transactions on 24*, 3 (1978), 384–386.

[5] BERNSTEIN, D. J., BUCHMANN, J., AND DAHMEN, E. *Post Quantum Cryptography*, 1st ed. Springer-Verlag, Berlin, Heidelberg, 2009.

[6] BERNSTEIN, D. J., LANGE, T., AND PETERS, C. Smaller decoding exponents: ball-collision decoding. In *Proceedings of the 31st annual conference on Advances in cryptology* (Berlin, Heidelberg, 2011), CRYPTO'11, Springer-Verlag, pp. 743–760.

[7] BISWAS, B., AND SENDRIER, N. McEliece Cryptosystem Implementation: Theory and Practice. In *Proceedings of the 2nd International Workshop on Post-Quantum Cryptography* (Berlin, Heidelberg, 2008), PQCrypto '08, Springer-Verlag, pp. 47–62.

[8] BOGDANOV, A., KHOVRATOVICH, D., AND RECHBERGER, C. Biclique cryptanalysis of the full AES. In *Proceedings of the 17th international conference on The Theory and Application of Cryptology and Information Security* (Berlin, Heidelberg, 2011), ASIACRYPT'11, Springer-Verlag, pp. 344–371.

[9] CANTEAUT, A., AND SENDRIER, N. Cryptanalysis of the Original McEliece Cryptosystem. In *Advances in Cryptology — ASIACRYPT'98*, K. Ohta and D. Pei, Eds., vol. 1514 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1998, pp. 187–199.

[10] DING, J., AND GOWER, J. E. Inoculating multivariate schemes against differential attacks. In *Proceedings of the 9th international conference on Theory and Practice of Public-Key Cryptography* (Berlin, Heidelberg, 2006), PKC'06, Springer-Verlag, pp. 290–301.

[11] FAUGÈRE, J.-C., AND PERRET, L. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 30–47.

[12] GROVER, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (New York, NY, USA, 1996), STOC '96, ACM, pp. 212–219.

[13] HOFFSTEIN, J., PIPHER, J., AND SILVERMAN, J. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, J. Buhler, Ed., vol. 1423 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1998, pp. 267–288.

[14] INTERNET ASSIGNED NUMBERS AUTHORITY (IANA). "Magic Numbers"for ISAKMP Protocol, oct 2000. Zuletzt Aktualisiert am 19.07.2012.

[15] INTERNET ASSIGNET NUMBERS AUTHORITY (IANA). Internet Key Exchange Version 2 (IKEv2) Parameters, jan 2005. Zuletzt Aktualisiert am 02.12.2014.

[16] KAUFMAN, C., HOFFMAN, P., NIR, Y., AND ERONEN, P. RFC5996 - Internet Key Exchange Protocol Version 2 (IKEv2), sep 2010.

[17] KENT, S., AND SEO, K. RFC4301 - Security Architecture for the Internet Protocol, dec 2005.

[18] KRAWCZYK, H., BELLARE, M., AND CANETTI, R. RFC2104 - HMAC: Keyed-Hashing for Message Authentication, feb 1997.

[19] LENSTRA, A. Unbelievable Security Matching AES Security Using Public Key Systems. In *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 67–86.

[20] LI, Y. X., DENG, R.-H., AND WANG, X.-M. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *Information Theory, IEEE Transactions on 40*, 1 (1994), 271–273.

[21] MATSUMOTO, T., AND IMAI, H. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Lecture Notes in Computer Science on Advances in Cryptology-EUROCRYPT'88* (New York, NY, USA, 1988), Springer-Verlag New York, Inc., pp. 419–453.

[22] MCELIECE, R. J. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report 44* (jan 1978), 114–116.

[23] MCGREW, D., AND HOFFMAN, P. RFC7321 - Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH), aug 2014.

[24] MISOCZKI, R., TILLICH, J.-P., SENDRIER, N., AND BARRETO, P. S. L. M. MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. Cryptology ePrint Archive, Report 2012/409, 2012.

[25] NIEDERREITER, H. Knapsack-type cryptosystems and algebraic coding theory. In *Problems of Control and Information Theory* (1986), vol. 15, pp. 159–166.

[26] PERSICHETTI, E. Compact McEliece keys based on Quasi-Dyadic Srivastava codes. Cryptology ePrint Archive, Report 2011/179, 2011.

[27] PFITZMANN, A. Sicherheit in Rechnernetzen: Mehrseitige Sicherheit in verteilten und durch verteilte Systeme, 2012.

[28] SCHILLER, J. RFC4307 - Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2), dec 2005.

[29] SHOR, P. W. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on Foundations of Computer Science* (nov 1994), IEEE Computer Society Press, pp. 124–134.

[30] SINGER, A. NTRU Cipher Suites for TLS - draft-ietf-tls-ntru-00, jul 2001.

[31] STALLINGS, W. *Cryptography and Network Security: Principles and Practice*, 5th ed. Prentice Hall Press, Upper Saddle River, NJ, USA, 2010.

[32] STEFFEN, A., WILLI, M., AND BRUNNER, T. strongSwan wiki, jun 2007.

[33] ZIEMBA, G., REED, D., AND TRAINA, P. RFC1858 - Security Considerations for IP Fragment Filtering, oct 1995.

[34] ZIMMER, E. Post-Quantum Kryptographie für IPsec. Technische Universität Dresden, jan 2014. Diplomarbeit.