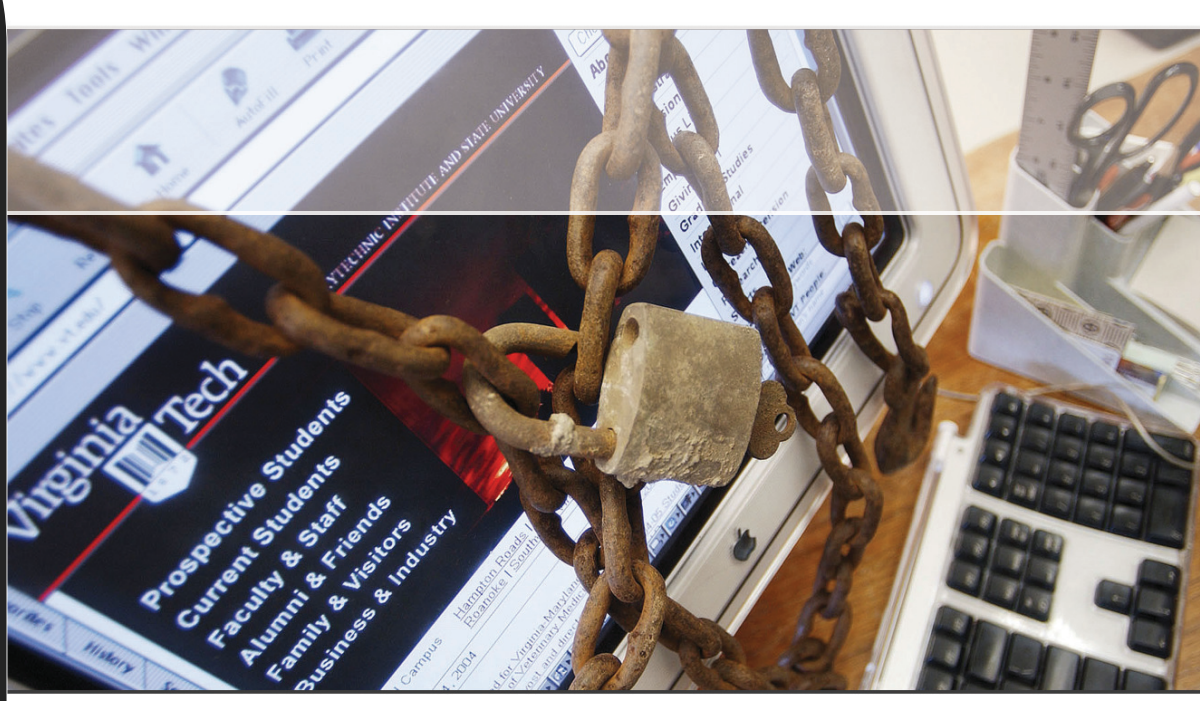

Bachelorarbeit Informatik

Multicast Security using IPsec

A lightweight solution using IKEv2

Cornel Eberle
Roman Federer

Frühlingssemester 2009

Betreut durch:
Prof. Dr. Andreas Steffen
Dipl. Ing. Martin Willi

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
INFORMATIK

strongSwan

Abstract

IPsec ist ein verbreitetes Protokoll zur Absicherung der Netzwerkkommunikation auf dem Network-Layer. Dabei wird Vertraulichkeit, Authentizität und Integrität der Daten gewährleistet. Das Internet Key Exchange Protokoll in Version 2 (IKEv2) wird für den Austausch der Schlüssel verwendet.

Bis heute existiert im IPsec-Standard keine Lösung für die Absicherung von IP Multicast Verkehr. Da es sich bei Multicast um Kommunikation zwischen mehr als zwei Teilnehmern handelt, muss ein gemeinsamer Schlüssel verteilt werden. Einige Ansätze einer IPsec Multicast Lösung wurden bereits entwickelt und im Internet publiziert. Diese sind jedoch entweder zu umfangreich, zu kompliziert in der Realisierung oder enthalten Schwachstellen.

Das Ziel dieses Projekts war die Entwicklung eines IPsec Multicast Protokolls, welches auf IKEv2 aufbaut. Die Lösung soll möglichst leichtgewichtig sein und so viel wie möglich vom Standard übernehmen. Zudem soll die Realisierung eines High-Level Protokollsimulators in der Programmiersprache C# die Funktionsweise des Protokolls anhand einer praktischen Umsetzung überprüfen.

Als Resultat dieser Arbeit gehen das Konzept einer IPsec Multicast Lösung sowie der darauf basierende Simulator hervor. Während der Implementierung des IPsec Multicast Prototyps konnten einige Details im Protokoll verfeinert werden. Schlussendlich bestätigt der Simulator, dass die theoretischen Überlegungen funktionieren und eine Integration in die bestehende IPsec Lösung strongSwan vollzogen werden kann.

Das Konzept basiert auf einem Group Controller and Key Server GCKS, der für die Verwaltung der Multicast-Gruppen und die Verteilung des Schlüsselmaterials zuständig ist. Um den Schlüssel für eine Multicast Gruppe zu erhalten, muss sich ein neuer Teilnehmer beim GCKS anmelden. Dazu sind zwei Nachrichtenpaare nötig, welche leicht abgeändert dem IKEv2 Standard entsprechen. Der erstellte Prototyp ist in der Lage eine IPsec Multicast Umgebung zu simulieren. Die Software enthält die Funktionalität für das Betreiben eines GCKS sowie für das Senden und Empfangen von Multicast-Daten. Verschiedene Szenarien lassen sich mit Hilfe des Simulators automatisch oder in manueller Form austesten.

Inhaltsverzeichnis

1 Einführung

Folgende Einführung wurde hauptsächlich dem Bericht „Multicast Anwendungen auf Basis von IPsec“ von E-Neumann [1] entnommen.

IP Multicast und IPsec (Internet Protocol Security) sind zwei etablierte Technologien mit unterschiedlichen Zielsetzungen: IP Multicast reduziert die notwendige Bandbreite beim Versand des gleichen Inhalts an viele Adressaten und IPsec sichert IP Datenströme ab. Die Kombination beider Technologien würde also den gesicherten Versand an viele Teilnehmer erlauben. Anwendungsszenarien wären z.B. vertrauliche Telefonkonferenzen oder Pay-TV. Je nach Sicht und Vorwissen des Lesers geht es in diesem Beitrag daher um die Absicherung von IP Multicast oder um die Erweiterung von IPsec.

1.1 Grundlagen IP Multicast

Bei traditionellem IP Unicast werden Pakete von einem Sender zu einem Empfänger transportiert. Dieses Verfahren benötigt insbesondere serverseitig sehr viel Bandbreite, wenn die gleichen Pakete an viele Empfänger gesendet werden müssen.

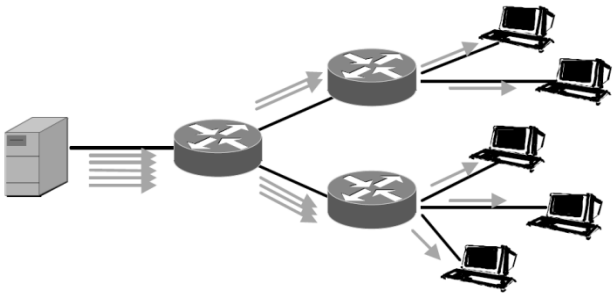

Abbildung 1: IP Unicast

IP Multicast reduziert die Netzlast, indem Pakete erst am letztmöglichen Knoten dupliziert werden.

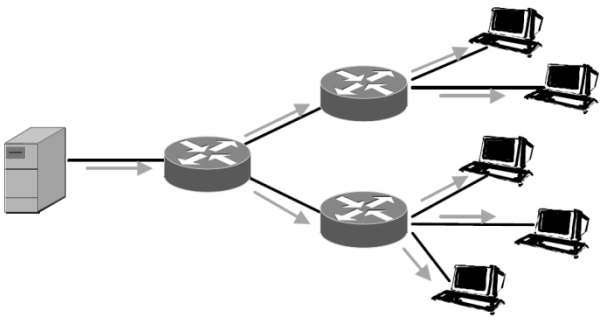

Abbildung 2: IP Multicast

Wie die Abbildung 2 bereits verdeutlicht gibt es bei IP Multicast drei verschiedene Rollen:

Sender

Der Sender verschickt ein Multicast Paket genau einmal. Die Besonderheit liegt in der Zieladresse, die das Paket als Multicast Paket kennzeichnet.

Empfänger

Damit ein Multicast Paket einen Empfänger erreicht, muss dem Netzwerk bekannt sein, dass der Empfänger an Paketen mit dieser Zieladresse interessiert ist. Der Empfänger muss dazu dem nächstgelegenen Router sein Interesse an Paketen mit einer bestimmten Multicast Adresse signalisieren.

Router

Die Router haben die komplizierte Aufgabe festzustellen, welche Pakete an welche Schnittstellen weitergeleitet werden müssen. Wie auch bei IP Unicast hängt die Entscheidung von der Zieladresse ab. Ein Multicast Router muss also pro Zieladresse speichern, an welchen Schnittstellen Empfänger angeschlossen sind. Gegenüber anderen Routern muss er sich aber selbst wie ein Empfänger verhalten.

Eigenschaften

Das Netz regelt die Verteilung der Pakete, die über IP Unicast hinausgehen. D.h. es werden keine Anforderungen an den Sender gestellt. Insbesondere muss der Sender das IP Multicast Protokoll IGMP [2] nicht implementieren.

Die Empfänger und alle dazwischen liegenden Router müssen IGMP implementieren, um den benachbarten Routern das Interesse an Multicast Paketen zu signalisieren. Wenn Teile des Netzwerkes kein Multicast unterstützen, besteht allerdings die Möglichkeit die Multicast und IGMP Pakete durch Unicast Tunnel zu senden.

Sender dürfen auch Empfänger sein und umgekehrt. So sind z.B. Konferenzsysteme realisierbar. Insbesondere darf es für einzelne IP Multicast Gruppen (entspricht einer IP Multicast Adresse) mehrere Sender geben.

Die Menge der Teilnehmer an einer Multicast Sitzung ist dynamisch. Während die Sitzung läuft, können Teilnehmer der Sitzung beitreten oder die Sitzung verlassen. Insbesondere ist die Menge der Teilnehmer zu Beginn der Sitzung nicht bekannt.

IP Multicast selbst sieht keine Empfangsbestätigung vor. Verbindungslose Protokolle wie UDP lassen sich über IP Multicast betreiben.

Die fehlende Kontrolle der Empfänger ist nicht nur ein Problem der Zuverlässigkeit der Übertragung. Eine weitere Folge ist auch die mangelnde Sicherheit. Prinzipiell kann der Sender nämlich auch die Gruppe der Empfänger nicht effektiv einschränken. Damit ist IP Multicast für Lauschangriffe noch anfälliger als IP Unicast.

Anwendungen

- Audio/Video Ausstrahlung: Fernsehen, Vorträge, Ansprachen, Radio, usw.
- Push Medien: News Ticker, Wettermeldungen, usw.
- Überwachung: Börsenkurse, Zustand von Netzwerkkomponenten, usw.
- Signale: Zeit, Standort mobiler Geräte, usw.

1.2 Grundlagen IPsec

IPsec ist das umfangreichste Sicherheitsprotokoll für IP. Kurz gesagt ist IPsec eine VPN-Technologie, die sichere Tunnel über unsichere Netze etabliert. IPsec stellt Mechanismen für alle gängigen Sicherheitsanforderungen bereit. Die Besonderheit von IPsec ist aber seine Flexibilität. Funktionalitäten lassen sich gemäss den Anforderungen aktivieren oder deaktivieren. Für die Verschlüsselung und Authentifizierung können sogar die Algorithmen ausgewählt werden.

Das Internet Key Exchange Protocol (IKEv2) wird von IPsec für den Schlüsselaustausch verwendet. IKEv2 wurde im Dezember 2005 standardisiert und soll einige Schwachstellen bzw. Probleme seines Vorgängers IKE korrigieren. Der Verbindungsaufbau wurde vereinfacht. Statt neun gibt es nur noch vier IKE-Meldungen. Insgesamt ist IKEv2 weniger komplex als sein Vorgänger, was die Implementierung erleichtert und die Sicherheit erhöht.

Anwendung

Üblicherweise wird IPsec verwendet, um private Netze über öffentliche Netze hinweg sicher zu verbinden. Dazu wird jedes private Netz über einen IPsec Gateway mit dem öffentlichen Netz verbunden. Die Gateways bauen bei Bedarf untereinander sichere Tunnel auf. Host Implementierungen sind ebenfalls möglich, kommen jedoch meist nur im privaten Bereich oder bei mobilen Nutzern zum Einsatz. Die Verwendung von Gateways hat den Vorteil, dass ausser den Gateways keine Komponente von den Tunneln wissen muss. D.h. die Verwendung von IPsec ist für den Sender, den Empfänger und für alle dazwischen liegenden Knoten (ausser den Gateways) transparent.

Kritikpunkte

Was die Sicherheit von IPsec betrifft, bestehen keine ernsthaften Bedenken. Allerdings gibt es andere Kritikpunkte. Erstens ist die Zahl der Tunnel unter Umständen sehr gross. Bei 20 Teilnehmern kann es zum Beispiel bis zu 190 Tunnel geben. Teilt man den Traffic je nach Sicherheitsanforderungen auf verschiedene Tunnel auf, kann die Zahl sogar noch deutlich grösser sein. Die Anforderungen an die Bedienbarkeit und Strukturierung der Management Software sind also sehr hoch.

Verbindungsaufbau

Die Konfiguration eines IPsec Gateways ähnelt in bestimmten Aspekten der einer Firewall. Je Quelladresse, Zieladresse, Schnittstelle, Protokoll, Port usw. wird definiert, ob das Paket verworfen oder weitergeleitet wird. Bei einem IPsec Gateway kommt als dritte Möglichkeit der Versand durch einen Tunnel in Frage. Für diesen Fall müssen die Parameter des Tunnels definiert werden. Der wichtigste Parameter ist natürlich die Adresse des Ziel-Gateways.

Wenn ein Paket durch einen Tunnel geschickt werden muss, der noch nicht besteht, nimmt der Gateway Kontakt zum entsprechenden Ziel-Gateway auf, um die zu verwendenden kryptographischen Algorithmen und Schlüssel auszuhandeln. Tunnel heissen in der Terminologie von IPsec Security Associations (SAs). In IKEv2 wird dabei zwischen zwei Typen von SAs unterschieden: Die von IKE für die Aushandlung der eigentlichen Kommunikationsparameter verwendeten IKE_SAs und die diesen zugeordneten CHILD_SAs mit den jeweils individuell ausgehandelten Kommunikationsparametern.

Ein wichtiger Punkt in der IKEv2-Terminologie sind die verteilten Rollen: Der Kommunikationspartner, der einen Meldungsaustausch initiiert, wird als *Initiator* bezeichnet. Der auf die Initialisierung antwortende Kommunikationspartner wird als *Responder* bezeichnet.

IKEv2 definiert vier Meldungstypen: IKE_SA_INIT für die Initialisierung einer IPsec-Verbindung, IKE_AUTH für die Authentifizierung der Kommunikationspartner, CREATE_CHILD_SA für

Schlüsseländerungen und die Erstellung zusätzlicher CHILD_SAs sowie INFORMATIONAL für den weiteren Informationsaustausch.

Mittels IKE_SA_INIT-Nachricht startet die (unverschlüsselte) Initialisierung einer IPsec-Verbindung. Dabei werden die verwendeten Algorithmen ausgehandelt und die zu verwendenden Schlüssel über einen Diffie-Hellmann-Schlüsselaustausch ausgetauscht.

Nach erfolgreicher Initialisierung der Kommunikationspartner durch IKE_SA_INIT wird der Meldungstyp IKE_AUTH für die Authentifizierung der Kommunikationspartner verwendet. Aus dem dabei durch den Diffie-Hellmann-Schlüsselaustausch ausgetauschten Schlüssel werden alle weiteren benötigten Schlüssel (z.B. für Verschlüsselungs- und Signaturalgorithmen) abgeleitet.

Der Austausch der IKE_AUTH-Meldungen dient der gegenseitigen Authentifizierung der Kommunikationspartner und dem Aushandeln einer ersten CHILD_SA für die folgende Kommunikation über die Protokolle Authentication-Header (AH) und/oder Encapsulation Security Payload (ESP).

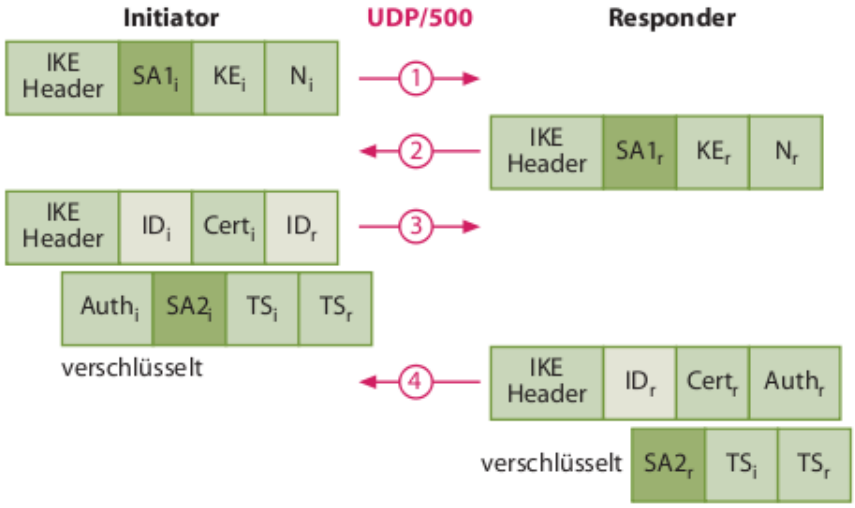

Abbildung 3: Für einen Verbindungsaufbau werden vier IKEv2-Meldungen benötigt

Die Aufteilung auf zwei verschiedene Typen von Tunnel ist einer der Gründe für die Komplexität von IPsec. Allerdings bietet die Trennung auch Vorteile. Erstens ist das Erstellen einer CHILD_SA mittels CREATE_CHILD_SA-Nachricht sehr schnell, weil keine Authentifizierung mehr nötig ist. Zweitens kann der Schlüssel, der während dem IKE_SA_INIT-Austausch für den äusseren Tunnel ausgehandelt wurde, lange Zeit benutzt werden, weil nur wenige Pakete damit verschlüsselt werden. Die Lebenszeit der IKE_SA kann also wesentlich höher sein als die Lebenszeit der CHILD_SAs. Schliesslich kann die IKE_SA auch „auf Verdacht" aufgebaut werden, um die Etablierung eines Tunnels bei Bedarf zu beschleunigen.

1.3 Verknüpfung der Themengebiete

Versucht man IPsec auf IP Multicast zu übertragen, so ergeben sich einige Probleme, die hier kurz erläutert werden sollen.

Schlüssel-Verteilung

Der Diffie-Hellman Algorithmus, den IPsec bei der Etablierung der IKE_SA verwendet, ermöglicht zwei Parteien die Einigung auf einen gemeinsamen Schlüssel, ohne dass dieser Schlüssel explizit übertragen werden muss. Dazu schickt jeder Teilnehmer einen Anteil. Beide Parteien können daraus den gemeinsamen Schlüssel errechnen, ein Zuhörer allerdings nicht. Leider lässt sich der Algorithmus nicht einfach von zwei auf mehr Parteien erweitern. Ausserdem sind bei einer Multicast Sitzung die Teilnehmer nicht im Vorfeld bekannt und verfügbar. Daher muss die Festlegung des gemeinsamen Schlüssels für Multicast IPsec auf andere Weise erfolgen. Dies ist das Hauptproblem von Multicast IPsec.

Tunnel Identifikation

IPsec verwendet einen sogenannten Security Parameters Index (SPI) zur Zuordnung von eingehenden Paketen zu einem Tunnel. Bei IPsec wird die SPI durch den Empfänger vergeben. Da die Empfänger zu Beginn der Sitzung nicht feststehen, ein einzelnes Paket mehrere Empfänger haben kann und zu Beginn der Sitzung vielleicht noch gar keine Empfänger vorhanden sind, kann bei Multicast IPsec die SPI nur durch den Sender oder durch eine unabhängige, zentrale Instanz vergeben werden.

Sequenz Nummern

Bei einer Replay Attacke sendet ein Angreifer ein mitgehörtes Paket. Dazu muss der Angreifer den Inhalt des Pakets nicht unbedingt verstanden haben. Es genügt schon, dass er die Wirkung des Pakets verstanden hat. Wenn er das Paket erneut schickt, kann er die Wirkung wiederholen. Zur Abwehr von Replay Attacken nummerieren die IPsec Protokolle die Pakete durch. Diese Sequenz Nummern sind aufsteigend, so dass alte Pakete erkannt und verworfen werden können. Bei Multicast Sitzungen kann es aber mehrere Sender geben. Für Multicast IPsec müssten die Sender die Sequenz Nummern untereinander synchronisieren. Das Problem der Sequenz Nummern tritt erst auf, wenn mehrere Sender beteiligt sind.

2 Aufgabenstellung

Einführung

Der heutige IPsec Standard bietet keine Unterstützung für die sichere Übertragung von IP Multicast Paketen. Das Hauptproblem, das gelöst werden muss, ist die Verteilung eines gemeinsamen Schlüssels an alle Mitglieder einer Multicast Gruppe. In dieser Arbeit soll ein Vorschlag für eine „*Group Key Distribution Protocol Extension*“ erarbeitet werden, die sich möglichst gut in Version 2 des Internet Key Exchange Protokolls (IKEv2) integrieren lässt. Für die Evaluation verschiedener Protokollvarianten soll ein High-Level Simulator in der Programmiersprache C# erstellt werden.

Aufgabenstellung

- Verstehen der Anforderungen an IPsec Multicast durch Studium der bestehenden RFCs und Internet Drafts.
- Verstehen der Funktionsweise des IKEv2 Protokolls am Beispiel der strongSwan Implementation.
- Erarbeiten einer IKEv2 Group Key Distribution Protocol Extension unter Berücksichtigung folgender Fragestellungen:
    - Ein Host soll einer beliebigen Anzahl von Multicast-Gruppen beitreten können.
    - Wie kann die Beitrittsberechtigung zu einer Multicast-Gruppe abgeklärt werden? Zentrale Policy-Datenbank auf dem Group Controller and Key Server (GCKS) und/oder verteilte Authorisierung via z.B. X.509 Attributzertifikate oder SAML Assertions?
    - Ein Host soll eine Multicast-Gruppe wieder verlassen können.
    - Wie soll das Rekeying und die damit verbundene Schlüsselverteilung durch den GCKS innerhalb einer Multicast-Gruppe gelöst werden?
    - Soll ein Rekeying periodisch oder jedes Mal beim Beitritt oder Austritt eines Gruppenmitglieds durchgeführt werden?
    - Darf jedes Gruppenmitglied Multicast-Pakete senden oder nur der GCKS?
    - **Optionale Fragestellung:** Wie kann die Berechtigung Multicast-Pakete zu versenden überwacht werden (z.B. via TESLA).

- Erstellen eines High-Level Protokoll-Simulators in der Programmiersprache C#
    - Es wird vorausgesetzt, dass ein Host, der einer Multicast-Gruppe beitreten will, sich schon via IKEv2 Standardmechanismen (RSA, EAP) gegenüber dem GCKS authentisiert hat.
    - Die Sicherung der IKE und ESP Kommunikation durch Verschlüsselung und MACs muss nicht implementiert werden, sondern soll nur symbolisch durch entsprechende Platzhalter-Payloads angedeutet werden.

Links

- strongSwan Projekt
  `http://www.strongswan.org/`
- Internet Key Exchange (IKEv2) Protocol, RFC 4306
  `http://tools.ietf.org/html/rfc4306`
- GKDP: Group Key Distribution Protocol
  `http://tools.ietf.org/html/draft-ietf-msec-gkdp-01`
- Multicast Security (MSEC) Group Key Management Architecture, RFC 4046
  `http://tools.ietf.org/html/rfc4046/`
- Multicast Extensions to the Security Architecture for the Internet Protocol, RFC 5374
  `http://tools.ietf.org/html/rfc5374`
- Timed Efficient Stream Loss-Tolerant Authentication (TESLA), RFC 4082
  `http://tools.ietf.org/html/rfc4082`
- The Use of TESLA in IPsec
  `http://tools.ietf.org/html/draft-dondeti-msec-IPsec-tesla-02`
- An Internet Attribute Certificate Profile for Authorization, RFC 3281
  `http://tools.ietf.org/html/rfc3281`
- Security Assertion Markup Language (SAML) v2.0
  `http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip`

Rapperswil, 25. Februar 2009

Prof. Dr. Andreas Steffen

3 Analyse

3.1 Anwendungsfälle

Für ein besseres Verständnis des Themas dieser Arbeit werden in diesem Abschnitt einige mögliche Anwendungen einer IPsec Multicast Lösung dargestellt. Diese lassen sich unterteilen in Infrastruktur- und Applikationsanwendungen.

3.1.1 Infrastrukturanwendungen

Für den Betrieb eines Netzwerks kommunizieren die einzelnen Netzwerkkomponenten untereinander. Diese Kommunikation beinhaltet unter anderem Informationen zum Verwalten der verschiedenen Netzwerkteilnehmer. Das Missbrauchen dieses Verkehrs ist ein bekanntes Angriffsziel. Gezieltes Verändern oder Wiedersenden solcher Pakete kann zu Störungen und Unterbrüchen im Netzwerk führen.

Ein möglicher Anwendungsfall von IPsec Multicast wäre das Schützen der Kommunikation von Netzwerkkomponenten, welche per Multicast untereinander kommunizieren. In der Abbildung 4 sendet ein Router geschützten Multicast Verkehr zu weiteren Routern im Netz. Sämtliche anderen Teilnehmer können diesen Verkehr weder lesen noch verändern.

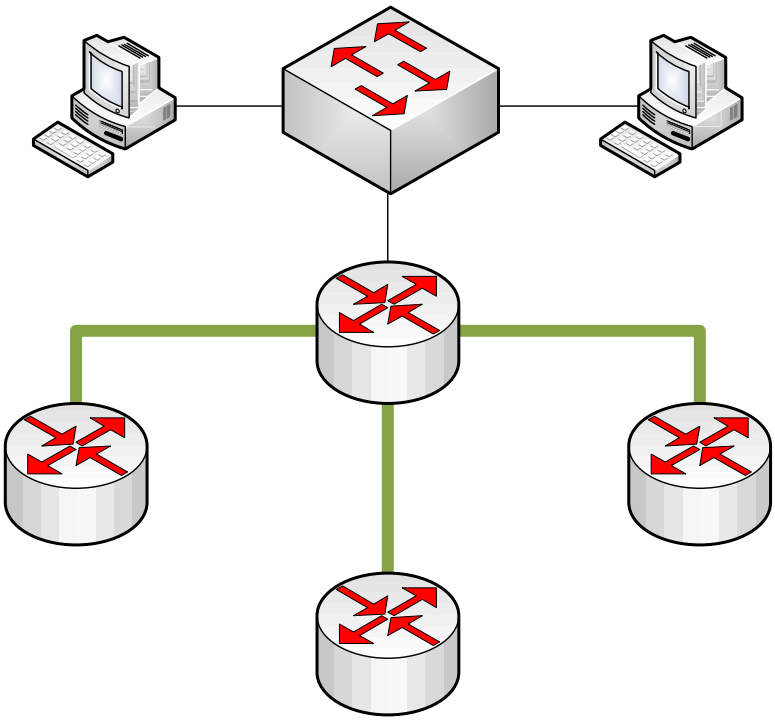

Abbildung 4: Geschützte Kommunikation in Gruppen

Für den Aufbau einer Multicastgruppe muss dem Netzwerk mitgeteilt werden, welcher Netzwerkteilnehmer welchen Multicastverkehr empfangen möchte. Dies geschieht mit dem Protokoll IGMP. Da die Kommunikation auch über Multicast-Gruppen stattfindet, könnte diese mit IPsec Multicast geschützt werden. Ein Internet-Draft zu diesem Thema [3] verdeutlicht diese Problematik. Dabei geht es nicht um den Datenverkehr, der geschützt werden muss, sondern lediglich um die An- und Abmeldung von einer Multicast-Gruppe. Für jede Multicast-Gruppe wird ein eigener Tunnel aufgebaut, welcher den Missbrauch von IGMP verhindert.

Abbildung 5: IPsec Multicast für IGMP

Die Abbildung 5 zeigt einen Endpunkt, der einer Multicastgruppe beitreten möchte. Dazu sendet er eine IGMP Nachricht an den nächsten Netzwerkknoten. Diese Nachricht ist durch IPsec Multicast geschützt. Die Anmeldung am Multicast-Verkehr wird im Netzwerk weitergeleitet, bis alle beteiligten Komponenten in Kenntnis gesetzt wurden.

3.1.2 Applikationsanwendungen

Auf Applikationsebene kann IPsec Multicast überall dort zum Einsatz kommen, wo Multicast-Verkehr gegen Abhören und/oder Verändern geschützt werden muss. Eine mögliche Anwendung wäre beispielsweise ein Newsticker-System.

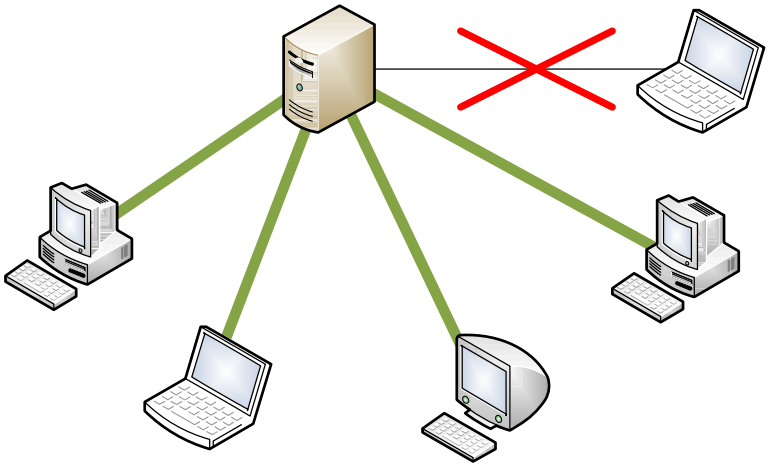

Abbildung 6: Anwendung IPsec Multicast auf Applikationsebene

Eine zentrale Stelle sendet Nachrichten über eine Multicast-Adresse. Möchte ein Teilnehmer diese Nachriten ebenfalls empfangen, so tritt er der Multicast-Gruppe bei. Die Abbildung 6 zeigt dieses Szenario auf. Dabei können die Teilneher, welche der IPsec Multicast Gruppe beigetreten sind (grüne Verbindung) den Verkehr empfangen und entschlüsseln. Ein Teilnehmer, der keinen Zutritt zur sicheren IPsec Multicast Gruppe erhält, kann den Verkehr zwar empfangen, aber den Schlüssel zur Entschlüsselung der Daten hat er nicht und die empfangenen Daten sind somit wertlos.

Nicht zum Einsatz kommen wird IPsec Multicast voraussichtlich im Multimedia Bereich. Für das Streaming von Echtzeitdaten existieren bereits funktionsfähige Protokolle wie RTP, bzw. SRTP.

3.2 Anforderungen

Im Rahmen dieser Bachelorarbeit wurde ein Konzept entwickelt, welches eine mögliche Lösung von IPsec Multicast darstellt. An diese Lösung wurden verschieden Anforderungen definiert. Dazu gehören Anforderungen zum Thema Sicherheit, Usability und Skalierbarkeit. Im Folgenden werden diese Anforderungen erläutert.

Mehrere Gruppen

Ein Host soll in der Lage sein, mehreren IPsec Multicast Gruppen beitreten zu können. Er kann zu einem beliebigen Zeitpunkt einer neuen Gruppe beitreten oder eine Gruppe verlassen. Die Anzahl der möglichen Gruppen soll dabei nicht limitiert sein.

Mehrere Sender

Beim normalen, ungesicherten IP Multicast können mehrere Sender auf derselben Multicast IP-Adresse senden. Bei IPsec Multicast soll dies ebenfalls der Falls sein. Eine Limitierung, dass nur ein einzelner Sender in jeder IPsec Multicast Gruppe existieren darf, soll es demnach nicht geben.

Autorisierung

Möchte ein neuer Host einer Gruppe beitreten, so muss die Berechtigung überprüft werden. Dazu müssen verschiedene Autorisierungsverfahren angewendet werden können. Eine Möglichkeit wäre eine zentrale Datenbank, in welcher die Beitrittsberechtigung nachgefragt werden könnte.

Die zu entwickelnde IPsec Multicast Lösung muss aber auch eine weitere Variante zur Autorisierung bereitstellen. Ein Host, der einer Gruppe beitreten möchte, soll seine Berechtigung mit einem Zertifikat beweisen können. Dieses Zertifikat wird während dem Anmeldeverfahren übermittelt und überprüft.

Verlassen einer Gruppe

Zu jedem beliebigen Zeitpunkt soll ein Host eine IPsec Multicast Gruppe wieder verlassen können. Nach einem Austritt erhält der Teilnehmer keinen weiteren Signalisierungsverkehr zu dieser Gruppe.

Schlüsselerneuerung

Die verwendeten Schlüssel zur Absicherung des Multicast-Verkehrs sollen zu bestimmten Zeitpunkten erneuert werden können. Falls ein Schlüssel erneuert wird, müssen alle Teilnehmer dieser Gruppe den neuen Schlüssel erhalten.

Kein Entschlüsseln alter Pakete

Wenn ein neuer Teilnehmer einer Gruppe beitritt, darf es diesem nicht möglich sein, ältere Pakete zu entschlüsseln. Aufgezeichnete Pakete dürfen also auch nach einem Beitritt zu einer IPsec Multicast Gruppe nicht zu entschlüsseln sein.

Kein Entschlüsseln nach Austritt

Ebenfalls soll nach einem Austritt eines Hosts aus einer Gruppe missbräuchlich aufgezeichneter Verkehr dieser Multicast-Gruppe nicht entschlüsselt werden können. Tritt also ein Host aus einer Gruppe aus, so kann er mit den im Nachhinein aufgezeichneten Paketen nichts anfangen.

3.3 Protokollbeschreibung

Die folgenden Abschnitte enthalten die Erläuterungen zum theoretischen Konzept der IPsec Multicast Lösung. Zu den einzelnen Teilproblemen wurden jeweils verschiedene Ansätze erarbeitet und zusammen mit den Betreuern evaluiert.

3.3.1 Beitritt Multicast-Gruppe

In einer herkömmlichen IKEv2 IPsec Verbindung handeln die zwei beteiligten Teilnehmer das benötigte Schlüsselmaterial aus. Da bei Multicast jedoch eine 1 zu n Kommunikation stattfindet, kann kein Aushandeln der Schlüssel durchgeführt werden. Aus diesem Grund wird eine zentrale Instanz verwendet, welche für das Verwalten der Gruppen-Teilnehmer und der Schlüssel zuständig ist. Diese Instanz wird als *Group Controller and Key Server (GCKS)* bezeichnet [4].

Der Beitritt zu einer Multicastgruppe ist an sich ein sehr umfangreiches Problemgebiet. Deshalb wurde dieser Bereich auf mehrere Teilprobleme aufgeteilt.

3.3.1.1 Lokalisierung des GCKS

Möchte ein Teilnehmer einer Gruppe beitreten, so meldet er sich beim GCKS und fordert das benötigte Schlüsselmaterial an. Wie findet ein potentieller Teilnehmer jedoch diesen GCKS? Zu dieser Fragestellung wurden drei mögliche Varianten untersucht.

Szenario 1: IP-Adresse im Absender des Multicast-Verkehrs

In diesem Szenario kennt der neue Teilnehmer lediglich die IP-Adresse der Multicast-Gruppe, welcher er beitreten möchte. Aus dem bereits vorhandenen Multicast-Traffic kann der Teilnehmer die IP des Senders bestimmen, der in diesem Fall auch gleich der GCKS ist.

Ablauf der Lokalisierung:

a. Mittels IGMP-Nachricht meldet sich der Teilnehmer für die Gruppe an.
b. Der verschlüsselte Multicast-Verkehr gelangt zum Teilnehmer. Dieser kann die IP-Adresse des Absender, welche der IP-Adresse des GCKS entspricht, bestimmen.
c. Der Teilnehmer kann sich beim GCKS anmelden und erhält das notwendige Schlüsselmaterial.

Dieses Szenario funktioniert nur, wenn der Sender ebenfalls der GCKS ist. Dieser Spezial-Fall tritt dann auf, wenn es sich um eine Single-Sender Multicast-Gruppe handelt.

Zudem gilt es zu beachten, dass es sich nicht zwingend um einen konstanten Multicast-Verkehr handeln muss. Dies kann zur Folge haben, dass es durchaus länger dauern kann, bis der Teilnehmer ein Multicast-Paket erhält und dadurch den GCKS lokalisieren kann.

Ein weiterer Nachteil liegt darin, dass die ersten empfangenen Pakete für den Teilnehmer nicht zu entschlüsseln sind. Zunächst muss er sich beim GCKS melden und das Schlüsselmaterial anfordern.

Szenario 2: IP-Adresse des GCKS bekannt

Eine weitere Möglichkeit ist, dass der neue Teilnehmer die IP-Adresse bereits kennt. Dem Teilnehmer wird aus Informationen von Dritten (Konfiguration, Benutzereingabe, usw.) die IP-Adresse des GCKS mitgeteilt.

Ablauf der Lokalisierung

a. Die IP-Adresse des GCKS ist dem Teilnehmer bekannt. Er meldet sich beim GCKS und fordert das gewünschte Schlüsselmaterial an.
b. Sobald die aktuellen Schlüsselinformationen in der Hand des Teilnehmers sind, setzt dieser eine IGMP Nachricht ab.
c. Der verschlüsselte Multicast-Verkehr gelangt zum Teilnehmer, welcher nun in der Lage ist die Daten zu entschlüsseln.

Da die IP-Adresse des GCKS dem Teilnehmer über Dritte mitgeteilt wird, ist dieser unabhängig vom Sender des Multicast-Verkehrs. Diese Eigenschaften sind Grundvoraussetzungen für eine Multi-Sender Architektur.

Szenario 3: IP-Adresse im Multicast-Verkehr verpackt

Dieses Szenario ist vergleichbar mit dem Szenario 1. Der Unterschied liegt, darin, dass der GCKS nicht zwingend Absender des Multicast-Verkehrs sein muss. Die IP-Adresse befindet sich in einem nicht-verschlüsselten Feld des Multicast-Payloads. Dieses Feld wird bei jedem Multicast-Paket gesendet.

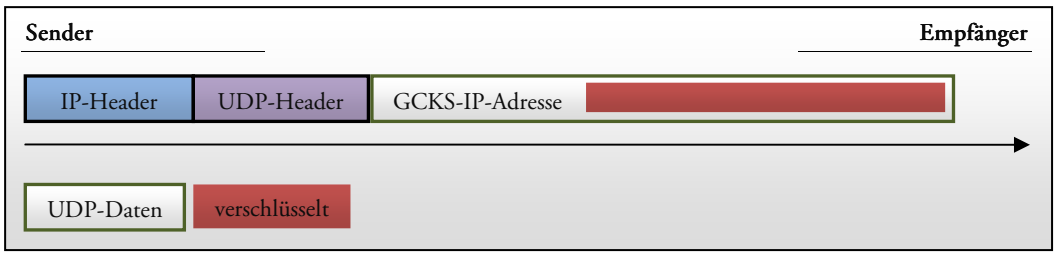

Abbildung 7: Payload aufgeteilt in IP-Adresse und verschlüsselten Multicast-Payload

Ablauf der Lokalisierung

a. Der Teilnehmer tritt mittels IGMP Nachricht der gewünschten Multicast-Gruppe bei.
b. Der Multicast-Verkehr erreicht nun den Teilnehmer, welcher aus dem unverschlüsselten Teil des Payloads die IP-Adresse des GCKS auslesen kann.
c. Der Teilnehmer kennt nun die IP-Adresse des GCKS. Sobald er sich bei diesem angemeldet hat, erhält er die notwendigen Schlüsselinformationen.

Durch die Aufteilung des Multicast-Verkehrs in zwei Bereiche weicht dieser von der Definition des Standards ab. Dies würde verschiedene Änderungen mit sich ziehen. Durch das permanente Verpacken der IP-Adresse des GCKS in den Multicast-Payload ergibt sich ein Overhead.

Die Vorteile dieses Szenarios liegen in der Flexibilität und der Möglichkeit eine Multi-Sender-Lösung zu realisieren.

Fazit Lokalisierungsszenarien

Nach einer Diskussion wurde im Review vom 17.03.2009 entschieden, dass im weiteren Verlauf des Projekts die Konzentration auf dem *Szenario 2: Teilnehmer kennt GCKS* liegt. Dieser Ansatz überzeugt durch die Einfachheit und lässt sich durch eine Konfigurationsdatei oder Benutzereingabe realisieren. Denkbar wäre auch ein Multicast-Service, welcher eine Art Verzeichnis darstellt. Bei diesem könnten die neuen Multicast-Teilnehmer die IP-Adresse des GCKS erfragen.

Der Nachteil des Szenario 1 liegt in der Beschränkung auf eine Single-Sender-Architektur. Das Szenario 3 würde eine Anpassung des Multicast-Standards beanspruchen. Auch wirkt sich der permanente Overhead des Multicast-Verkehrs negativ auf die Performance aus.

3.3.1.2 Aufsetzen einer GROUP_SA

Analog zu einer herkömmlichen IKEv2 SA beschreibt eine GROUP_SA die Parameter einer gesicherten Multicast-Gruppe. Darin wird festgelegt, welche Verschlüsselungs- und Integritätsverfahren zum Einsatz kommen.

Internet Draft: GKDP

Der Internet Draft *Group Key Distribution Protocol GKDP* [5] enthält einige Vorschläge zum Thema Multicast IPsec. Die darin enthaltenen Ideen wurden ausgiebig überprüft und diskutiert.

Das Projektteam zusammen mit den Betreuern ist jedoch zum Entschluss gekommen, dass ein eigener Vorschlag ausgearbeitet werden soll. Dieser setzt sich als Ziel, möglichst viel vom bestehenden IKEv2 Standard zu übernehmen. Die vorhandenen Nachrichten sollen leicht angepasst werden, sodass sie für das Erstellen und Verwalten einer GROUP_SA verwendet werden können. Zudem soll dieser Bericht konkretere Informationen über die Nachrichtenkommunikation enthalten.

Aufsetzen der SA

Für den Ablauf des Aufsetzens einer GROUP_SA wurde versucht, sich möglichst nahe an den Standard IKEv2 anzulehnen.

Nachdem der Teilnehmer über die Konfiguration oder eine Benutzereingabe herausgefunden hat, wer der zuständige GCKS für die Multicast-Gruppe ist, baut er zu diesem eine IKE_SA auf. Dabei kommt das herkömmliche IKEv2 Meldungspaar IKE_SA_INIT Request/Response zum Einsatz.

Die vorhandene IKE_SA muss anschliessend authentisiert und eine GROUP_SA für den Multicast-Verkehr hergestellt werden. Dies wird mit einem modifizierten IKE_AUTH Request/Response Paar erreicht. Um dem GCKS mitteilen zu können, dass es sich nicht um eine normale SA, sondern um eine GROUP_SA handelt, wird ein GROUP_SA Notify-Payload hinzugefügt.

Der neue Teilnehmer sendet seine SA-Vorschläge und die Traffic Selectors an den GCKS. In den Security Association-Vorschlägen sind die vom Teilnehmer unterstützten Algorithmen für die Verschlüsselung, Integrität wie auch die Diffie-Hellman Gruppe und die Pseudo Random Funktion enthalten. Die Traffic Selectors enthalten die Informationen über den gewünschten Multicast-Verkehr und dessen sendenden Teilnehmer.

Erhält der GCKS den IKE_AUTH Request, kann er aus den Traffic Selectors eindeutig die Gruppe bestimmen, für welche der Request gedacht ist. Nach erfolgreicher Autorisierung (siehe Abschnitt *3.3.1.3 Autorisierung* eines Teilnehmers) sendet der GCKS seine Auswahl aus den SA-Vorschlägen und die Traffic Selectors zur Komplettierung der GROUP_SA zurück. Ebenfalls ist in der Response das Schlüsselmaterial für die Multicast-Gruppe enthalten. Dieses wird in Form eines Notify-Payloads übermittelt.

Folgendes Detail ist noch anzumerken: Wenn der Initiator im IKE_AUTH Request seine SAs sendet, sind diese nicht als Vorschlag zu betrachten. Da es sich um eine GROUP_SA handelt, die erstellt werden soll und diese für mehrere Teilnehmer gültig sein muss, kann die SA nicht ausgehandelt werden. Es ist aber durchaus nützlich, dass der neue Teilnehmer seine möglichen SAs mitteilt und der GCKS diese dann bestätigt, bzw. mit einer NO_PROPOSAL_CHOSEN Notify verweigert.

3.3.1.3 Autorisierung eines Teilnehmers

Die Autorisierung eines neuen Gruppen-Teilnehmers kann auf verschiedene Arten erfolgen. Dabei sind zwei Grundsätzliche Fälle getrennt zu betrachten:

Szenario 1: Der GCKS ist selbst in der Lage den Teilnehmer zu autorisieren.

Szenario 2: Der GCKS benötigt zusätzliche Informationen vom Teilnehmer um diesen autorisieren zu können.

Szenario 1: GCKS autorisiert eigenständig

Der Teilnehmer initiiert bekanntlich zu Beginn eine IKE_SA mit dem GCKS. Dadurch kennt der GCKS die Identität des neuen Teilnehmers und hat diese authentisiert.

Wenn nun der Teilnehmer mittels IKE_AUTH Request eine GROUP_SA aufbauen möchte, kann der GCKS eigenständig den Teilnehmer autorisieren. Dazu sind erneut verschiedene Arten der Autorisierung denkbar. Beispielsweise kann es sich dabei um eine lokale oder entfernte Policy-Datenbank handeln. Es können aber auch Services zum Einsatz kommen, bei welchen der GCKS sich bei weiteren Komponenten beispielsweise eine SAML-Assertion oder ein Attributzertifikat besorgt.

Szenario 2: GCKS benötigt weitere Informationen

Auch in dieser Variante ist dem GCKS die Identität des neuen Teilnehmers bekannt. Der Unterschied besteht darin, dass der GCKS die Autorisierung aufgrund zusätzlicher Informationen durchführt.

Der Teilnehmer sendet diese zusätzlich benötigten Informationen bereits im IKE_AUTH Request. Dabei handelt es sich um zusätzliche Zertifikate, wie beispielsweise ein Attributzertifikat oder eine SAML-Assertion. Dadurch, dass diese Informationen bereits im IKE_AUTH Request enthalten sind, müssen nicht zusätzliche Nachrichten für die Autorisierung versandt werden.

SAML-Assertions werden als übliche Zertifikat-Payloads gesendet. Dazu muss lediglich ein neuer Zertifikatstyp für SAML-Assertions definiert werden.

Fazit Autorisierung der Teilnehmer

Für die Implementierung werden beide Szenarien berücksichtigt. Auf Grund der Komplexität von SAML-Assertions und Attributzertifikaten wurde die Autorisierung nur auf symbolischer Ebene implementiert.

3.3.2 NAT – Network Address Translation

Beim herkömmlichen IKEv2 wird im IKE_SA_INIT Nachrichtenaustausch die effektive Absender-Adresse eingefügt. Der Empfänger kann dann die eingefügte Adresse mit der Absender-Adresse des übermittelten Pakets vergleichen und erkennt so eine mögliche NAT-Situation.

Die NAT-Probleme, die bei Multicast IPsec auftreten, sind grundsätzlich dieselben, die auch bei ungeschütztem Multicast-Verkehr auftreten. Deshalb wird auf diese Probleme nicht speziell eingegangen.

3.3.3 Security Association Architektur

3.3.3.1 Anzahl Security Associations

Die Anzahl der verwendeten Security Associations innerhalb einer Infrastruktur hängt davon ab, ob für den Austausch der Rekey-Informationen für die GROUP_SAs eine eigene SA (REKEY_SA) verwendet wird.

Bei der Verwendung einer REKEY_SA besteht die Möglichkeit, die Rekey-Informationen mittels Multicast an die Teilnehmer zu verteilen. Dies würde den Kommunikationsaufwand minimieren.

Möchte die Anzahl verschiedener SAs und somit auch der Administrationsaufwand so gering als möglich gehalten werden, können die neuen Schlüsselinformationen beim Rekeying der GROUP_SAs über die IKE_SA übermittelt werden. Dabei ist zu beachten, dass alle Teilnehmer per Unicast über die aktuellen Schlüssel informiert werden müssen.

Die folgenden Ausführungen lehnen sich an den Draft zum *Group Key Distribution Protocol* [5] wie auch an den RFC zu *Multicast Security MSEC* [4] an.

Szenario 1: Schlüsselverteilung per Unicast REKEY_SA

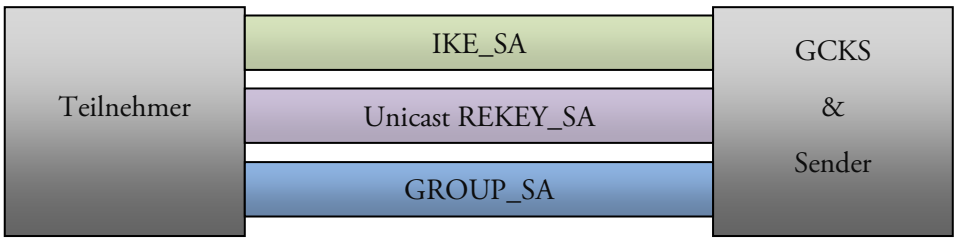

Abbildung 8: Infrastruktur mit einer IKE_SA, einer Unicast REKEY_SA und GROUP_SAs

Bei der Verwendung einer Unicast REKEY_SA besteht die Möglichkeit, jedem Teilnehmer einen spezifischen Key Encryption Key (KEK) auszuhändigen. Der Key Encryption Key bezeichnet den

Schlüssel der REKEY_SA. Der Schlüssel der GROUP_SA, Transport Encryption Key TEK genannt, ist jedoch gemein.

Der oben beschriebene Aufbau würde folgende **Eigenschaften** mit sich ziehen:

Unicast-Verteilung: Die Verteilung des TEKs wird mittels Unicast durchgeführt. Der GCKS weiss, welche Teilnehmer sich in der Gruppe befinden und kann einen sicheren Austausch garantieren.

Austritt: Da für jeden Teilnehmer eine eigene REKEY_SA besteht, kann der Austritt von Gruppenmitgliedern korrekt geregelt werden. Verlässt ein Teilnehmer die Gruppe, kann die REKEY_SA des Teilnehmers gelöscht und der TEK geändert werden. Danach hat der ausgetretene Benutzer keine Möglichkeit mehr, den Verkehr der GROUP_SA zu entschlüsseln.

Kein Multicast: Die individuellen KEKs verunmöglichen eine Verteilung der TEKs per Multicast.

Ablauf beim Rekeying

a. Der GCKS muss das neue Schlüsselmaterial (TEK bzw. KEK) an die Teilnehmer verteilen:
   - ***KEK:*** Da es sich bei der REKEY_SA um eine CHILD_SA (IPsec SA) handelt, kann ein Rekeying gemäss IKEv2-Standard [6] durchgeführt werden.
   - ***TEK:*** Der GCKS informiert die Teilnehmer einzeln mittels REKEY_SA über den neu zu verwendenden TEK. Die Teilnehmer setzen die neue GROUP_SA auf, um den Verkehr weiterhin entschlüsseln zu können.
b. Konnte die neue SA (REKEY_SA bzw. GROUP_SA) korrekt aufgebaut werden, wird der Datenverkehr über diese gesandt. Daraufhin kann die alte SA deaktiviert und gelöscht werden.

Bewertung des Szenarios

Die Verwendung einer REKEY_SA für die Verteilung des Schlüsselmaterials bringt keine markanten Vorteile mit sich. Durch die Verwendung einer Unicast SA für die Schlüsselverteilung, kann der GCKS Buch führen, wer das neue Schlüsselmaterial bereits erhalten hat und kann dadurch den genauen Umschaltzeitpunkt definieren.

Szenario 2: Schlüsselverteilung per Multicast REKEY_SA

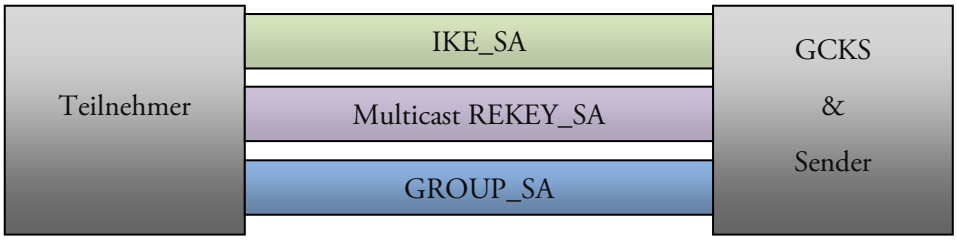

Abbildung 9: Infrastruktur mit einer IKE_SA, einer Multicast REKEY_SA und GROUP_SAs

Falls für die REKEY_SA gemeinsame Key Encryption Keys verwendet werden, kann der Datenverkehr der REKEY_SA per Multicast verteilt werden. Die REKEY_SA wird für den Schlüsselaustausch des Key Encryption Keys (KEK) wie auch des Transport Encryption Keys (TEK) verwendet.

Dabei müssen folgende ***Punkte*** beachtet werden:

Multicastverteilung: Da eine gemeinsame REKEY_SA für alle Teilnehmer verwendet wird, können der KEK und TEK mittels Multicast verteilt werden. Die REKEY_SA ist vergleichbar mit der Funktion einer GROUP_SA.

Administration: Da der GCKS nicht alle Teilnehmer kennen und ihnen explizit die Informationen zukommen lassen muss, verringert sich der Administrationsaufwand im Vergleich zur Variante 1.

Empfangssicherheit: Da bei der Verteilung mittels Multicast keine Empfangsbestätigung versandt werden kann, wird der GCKS über den Empfang der Schlüsselinformationen (KEK bzw. TEK) nicht informiert. Der GCKS weiss somit nicht, ob alle Teilnehmer die neuen Informationen erhalten haben und kann keinen exakten Umschaltzeitpunkt bestimmen.

Austritt: Tritt ein Mitglied aus der Gruppe aus, kann es die Rekey-Informationen immer noch entschlüsseln, da sie über die alte REKEY_SA verteilt werden. Dies hat zur Folge, dass ein Schlüsselwechsel nach einem Austritt keinen Nutzen mit sich bringt.

Ablauf beim Rekeying

a. Der GCKS muss das neue Schlüsselmaterial (TEK bzw. KEK) an die Teilnehmer verteilen:
   - *KEK und TEK:* Der GCKS informiert die Teilnehmer mittels REKEY_SA über den neuen KEK bzw. TEK. Die Teilnehmer setzen die neue REKEY_SA bzw. GROUP_SA auf.

b. Da keine Bestätigung über den Empfang des neuen Schlüsselmaterials versandt wird, muss eine Wartezeit definiert werden. Diese legt fest, wie lange den Teilnehmern Zeit gegeben wird, bis auf die neue SA gewechselt wird.

c. Nach Ablauf dieser Wartezeit wird der Wechsel auf die neue SA durchgeführt und die alte SA wird gelöscht.

Alternatives Szenario

Der Key Encryption Key (KEK) kann auch per Unicast über die IKE_SA an die Teilnehmer verteilt werden.

Bewertung des Szenarios

Ein grosser Nachteil dieses Szenarios ist, dass ein Teilnehmer nach dem Verlassen einer Gruppe die Multicast-Daten weiterhin entschlüsseln kann. Ist er einmal der Gruppe beigetreten, erhält er nämlich stets die neuen Schlüsselinformationen und kann dadurch den Multicast-Verkehr entschlüsseln. Dieses Problem könnte durch die Verwendung des alternativen Szenarios ein wenig eingeschränkt, aber nicht eliminiert werden.

Ein weiterer Lösungsansatz liegt in der Verwendung von sogenannten Group Key Management Algorithmen (siehe *3.3.3.2 Schlüsselgenerierung*).

Szenario 3: Schlüsselverteilung per Unicast IKE_SA

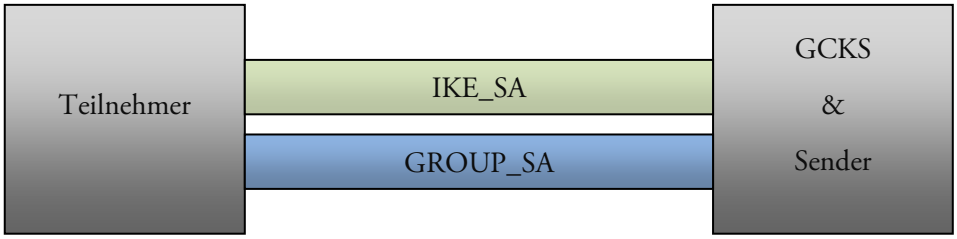

Abbildung 10: Infrastruktur mit Verwendung einer IKE_SA und GROUP_SAs

Wird für die Verteilung des GROUP_SA Schlüssels (TEK) keine eigene REKEY_SA verwendet, werden diese Informationen per Unicast über die IKE_SA an die Teilnehmer der Multicastgruppe verteilt. Diese Variante ist sehr nahe am IKEv2 Standard.

Folgende **Eigenschaften** müssen dabei beachtet werden:

Administration: Der Teilnehmer bzw. der GCKS muss sich nur um die IKE_SA und die verwendeten GROUP_SAs kümmern. Die Verwaltung einer speziellen RE-KEY_SA fällt weg.

Multicast: Da die neuen Transport Encryption Keys TEKs über die IKE_SA übermittelt werden, müssen/können diese per Unicast versandt werden. Dies hat den Vorteil, dass jeder Teilnehmer den Empfang des neuen TEK bestätigen kann und der GCKS dadurch Bescheid weiss, wann alle Teilnehmer die Meldungen empfangen haben. Dadurch kann ein lückenloser Empfang des Multicast-Verkehrs garantiert werden.

Austritt: Tritt ein Mitglied aus einer Multicast-Gruppe aus, so kann der Austritt sauber geregelt werden. Der austretende Subscriber kann garantiert vom Empfang der neuen Schlüssel ausgeschlossen werden.

Ablauf beim Rekeying

Das Rekeying der IKE_SA wird gemäss IKEv2-Standard [6] gehandhabt. Das Rekeying der GROUP_SA hat folgenden Ablauf.

a. Wechselt der Schlüssel (TEK) der GROUP_SA, wird dieser über die IKE_SA an die Multicast-Teilnehmer verteilt.
b. Alle Teilnehmer etablieren die neue GROUP_SA und bestätigen den Empfang des TEK.
c. Sobald der GCKS von allen Gruppenmitgliedern die Empfangsbestätigung des neuen Schlüsselmaterials erhalten hat, kann der Wechsel auf die neue GROUP_SA vorgenommen werden. Die alte GROUP_SA kann danach gelöscht werden.

Bewertung des Szenarios

Diese Variante hält den Administrationsaufwand so gering als möglich. Ebenfalls werden die Key Encryption Keys nach dem Wechsel der GROUP_SA nicht an ehemalige Teilnehmer verteilt.

Fazit Anzahl Security Associations

Aufgrund der obigen Erläuterungen setzt sich das *Szenario 3: Schlüsselverteilung per Unicast IKE_SA* durch. Der Hauptgrund für den Entscheid liegt in der korrekt geregelten Handhabung der Austritte der Multicast-Teilnehmer. Das Szenario 1 bringt dabei keinen wesentlichen Vorteil, wodurch der Mehraufwand der Administration hätte begründet werden können. Das Szenario 3 besticht nebenbei auch durch die Nähe zum bereits etablierten IKEv2 Standard.

3.3.3.2 Schlüsselgenerierung

Nach Ablauf der Lebenszeit einer GROUP_SA bzw. nach einem speziellen Ereignis (Ein-, Austritt eines Teilnehmers) muss neues Schlüsselmaterial generiert werden. Dabei kann das neue Schlüsselmaterial vom Server generiert oder mittels Algorithmus durch die Teilnehmer ausgehandelt werden.

Serverbasierte Schlüsselgenerierung

Der neue Schlüssel (Transport Encryption Key TEK) wird unabhängig von den Teilnehmern durch den GCKS generiert.

- ✓ Der Server kann ohne auf andere Teilnehmer Rücksicht nehmen zu müssen das neue Schlüsselmaterial generieren.
- ⊗ Bei grosser Anzahl von Teilnehmer, Gruppen und Ereignissen (z. B. Ein-, Austritte), die ein Rekeying nach sich ziehen, wird der Aufwand für den GCKS ziemlich gross.

Algorithmische Schlüsselgenerierung

Mittels sogenanntem Group Key Management Algorithmus GKMA können die Teilnehmer die neuen Schlüssel (TEK) aushandeln. Dazu eignen sich Algorithmen wie LKH [7], OFT [8] oder MARKS [9]. Auf deren genaue Funktionsweise wird nicht eingegangen.

- ✓ GCKS muss sich nicht um die Generierung der Schlüssel kümmern und kann dadurch entlastet werden.
- ⊗ Die Implementierung eines Group Key Management Algorithmus ist sehr aufwändig und komplex.

Fazit Schlüsselgenerierung

Aufgrund der aufwändigen Implementierung eines Group Key Management Algorithmus und der Entscheidung für das Szenario 3 aus dem Abschnitt *3.3.3.1 Anzahl Security Associations* ist der GCKS für die Generierung des neuen TEK verantwortlich.

3.3.3.3 Security Association Übergang

IKE_SA-Übergang

Da die IKE_SA gemäss IKEv2-Standard [6] implementiert wird, können die Informationen bezüglich des SA-Übergangs daraus entnommen werden.

GROUP_SA-Übergang

Beim Übergang von der einen GROUP_SA zu dessen Nachfolger ist es wichtig, dass die Multicast-Pakete stets entschlüsselt werden können. Aus diesem Grund wird die GROUP_SA vom GCKS zuerst an alle empfangenden Teilnehmer publiziert. Haben alle den Aufbau der neuen GROUP_SA bestätigt wird diese an den Sender weitergeleitet. Sobald der Sender die neue GROUP_SA erhalten

hat, beginnt er über diese zu senden. Hat der GCKS die Bestätigung des Senders erhalten, kann auf allen Teilnehmern die alte GROUP_SA gelöscht werden.

3.3.3.4 Schlüsselverteilung

Die neuen Schlüsselinformationen können mittels Unicast oder auch mittels Multicast an die Teilnehmer verteilt werden.

Unicast-Verteilung

Möchte der GCKS die Schlüsselinformationen mittels Unicast verteilen, so muss er Buch über alle Gruppen-Mitglieder führen. Damit die Schlüsselinformationen sicher übertragen werden, wird eine SA benötigt. Aufgrund der Entscheidung für die Architektur gemäss *Szenario 3: Schlüsselverteilung per Unicast IKE_SA* wird dazu die IKE_SA verwendet.

Eigenschaften der Unicast-Verteilung

- ✓ Der GCKS weiss genau, zu welchem Zeitpunkt alle Benutzer die neuen Schlüsselinformationen besitzen und kann somit einen optimalen SA-Wechsel durchführen.
- ✓ Die Schlüssel werden nur an die Teilnehmer gesandt, die auch den Verkehr empfangen dürfen. Beispielsweise kann ein Teilnehmer nach einem Austritt und einem darauffolgenden Schlüsselwechsel das neue Schlüsselmaterial nicht mehr empfangen und somit den Multicast-Verkehr nicht mehr entschlüsseln.
- ⊗ Der Administrationsaufwand für den GCKS ist ziemlich gross. Er muss über die aktuellen Teilnehmerlisten der verschiedenen Gruppen Bescheid wissen.

Unicast-Verteilung der Schlüsselinformationen eignet sich vor allem, wenn nach einem Austritt eines Teilnehmers ein Rekeying durchgeführt wird und der neue Traffic nicht mehr entschlüsselt werden darf. Ebenfalls ist die Verteilung per Unicast bei kleinen Teilnehmergruppen optimal.

Multicast-Verteilung

Wird für die Übermittlung des neuen Schlüsselmaterials eine REKEY_SA verwendet, so können diese mittels Multicast an die Teilnehmer verteilt werden. Dabei wird das Schlüsselmaterial (TEK) mit dem Key Encryption Key KEK verschlüsselt und an alle Gruppenteilnehmer versandt.

Eigenschaften der Multicast-Verteilung

- ✓ Da der GCKS keine Informationen über die einzelnen Teilnehmer führen muss, ist der administrative Aufwand geringer.
- ⊗ Der GCKS weiss bei einem Rekeying nicht, wann alle Teilnehmer den neuen Schlüssel besitzen. Dazu muss der GCKS eine Annahme treffen. Hat ein Teilnehmer den Schlüsselwechsel nicht mitbekommen, muss er die Möglichkeit haben, den aktuellen Key anzufordern.
- ⊗ Der Austritt eines Teilnehmers kann nicht sauber gelöst werden. Bei einem Rekeying nach dem Austritt eines Teilnehmers kann dieser die neuen Schlüsselinformationen immer noch erhalten und mit den alten Schlüsselinformationen entschlüsseln.

Bei sehr grossen Teilnehmergruppen eignet sich die Verteilung mittels Multicast. Voraussetzung für die Verteilung per Multicast ist, dass es unproblematisch ist, wenn ein Teilnehmer ein Rekeying nicht mitbekommt und dadurch das aktuelle Schlüsselmaterial erneut anfordern muss. Des Weiteren gibt es keinen wirklichen Austritt aus einer Gruppe. Der Teilnehmer kann nach einem Austritt den Multicast-Verkehr und somit die neuen Schlüssel weiterhin empfangen und entschlüsseln.

Fazit Schlüsselverteilung

Aufgrund des Verzichts der REKEY_SA werden die Rekey-Informationen mit den neuen Schlüsselinformationen für die IKE_SA und/oder die GROUP_SA über die IKE_SA mittels Unicast verteilt.

3.3.4 Adressierung

3.3.4.1 Traffic Selectors

Für den Aufbau der Security Associations werden sogenannte Traffic Selectors benötigt. Dabei können die Traffic Selectors TSr und TSi nicht wie in IKEv2 verwendet werden. Dies hängt damit zusammen, dass sich der Initiator auf der Seite des Teilnehmers wie auch auf der Seite des GCKS befinden kann, aber stets die Multicastadresse der Empfänger und die Unicastadresse des Senders übermittelt werden muss. Aus diesem Grunde definieren wir hierzu zwei neue Payloads mit den Namen *Traffic Selector Multicast TSm* für die Empfänger und *Traffic Selector Multicast Sender TSms* für den Sender. Ebenfalls werden in den Traffic Selectors die jeweiligen verwendeten Ports definiert.

Weiter ist zu beachten dass die verwendete Subnetzmaske jeweils 255.255.255.255 sein muss, da es sich nicht um einen Range sondern nur um eine definierte IP-Adresse handelt.

Diese Bedingungen führen beispielsweise zu folgenden Zuweisungen:

TSm	TS Multicast	IP-Adresse: 224.1.2.3/32	Port: 65000
TSms	TS Multicast Sender	IP-Adresse: 192.168.1.1/32	Port: 65821

Tabelle 1: Beschreibung der Traffic Selectors

3.3.4.2 Header der Nachrichten

Der Header der IKE-Nachrichten basiert auf den Spezifikationen des IKEv2 Standards. Dabei ist vor allem wichtig, dass die ersten zwei Werte aus den Security Parameter Indexen (SPI) bestehen. Die ersten 64 Bit bestehen aus dem SPI des IKE_SA Initiators und die zweiten 64 Bit aus dem SPI des Responders.

Wird eine neue SA angefordert, befindet sich der neue Security Parameter Index im Security Association Payload. Dies gilt für das Erstellen einer neuen IKE_SA wie auch einer neuen GROUP_SA.

Identifiziert werden die IKE_SAs durch den SPI des Initiators wie auch durch den SPI des Responders. Die IKE_SA wird für bidirektionalen Verkehr verwendet. Im Gegensatz dazu wird die GROUP_SA nur durch ein SPI identifiziert, welche durch den GCKS festgelegt wird. Ein weiterer Unterschied im Bezug zur IKE_SA liegt darin, dass eine GROUP_SA nur unidirektional ist.

Für detailierte Beschreibungen wird auf den Abschnitt 3 des IKEv2 RFCs [6] verwiesen.

3.3.4.3 Multicast-Verkehr

Der Multicast-Verkehr wird gewöhnlich adressiert und verschickt. D.h. die Empfängeradresse ist die jeweilige Multicastadresse und die Absenderadresse ist die IP-Adresse des Senders. Damit nicht jeder die Nutzdaten lesen kann, ist der Payload authentifiziert und verschlüsselt.

Der Paketaufbau sieht folgendermassen aus:

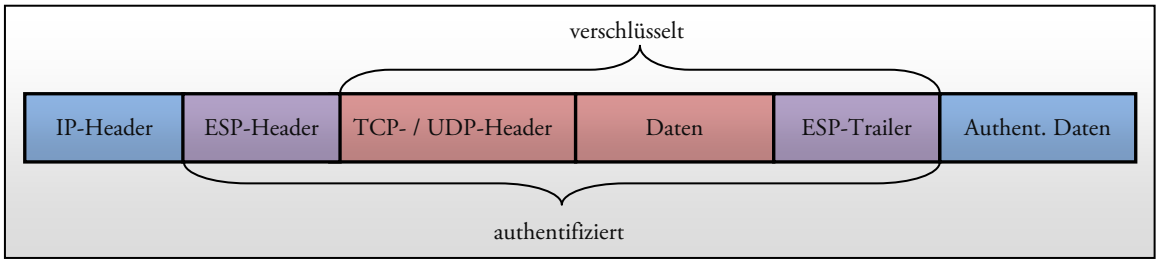

Abbildung 11: ESP im Transportmodus

3.3.5 Austritt Multicast-Gruppe

Möchte ein Teilnehmer aus einer Multicast-Gruppe austreten, muss er den GCKS darüber informieren. Dies geschieht mit sogenannten Informational Nachrichten über die IKE_SA. Hat der GCKS eine Delete Nachricht empfangen, entfernt er den Teilnehmer aus der Gruppe und bestätigt den Empfang. Danach führt er gegebenenfalls ein Rekeying durch, damit das ausgetretene Mitglied den Multicast-Verkehr nicht mehr entschlüsseln kann.

Verabschiedet sich ein Teilnehmer still aus den Multicastgruppen, so stellt der GCKS fest, dass der Teilnehmer über die IKE_SA nicht mehr erreichbar ist. Danach schliesst er die IKE_SA zum Teilnehmer und löscht den Teilnehmer aus allen Gruppen. Dieser Mechanismus ist bereits im IKEv2 Standard [6] integriert.

Der GCKS kann mittels Timeout des Rekeying oder durch ein separates Timeout feststellen, ob ein Teilnehmer noch aktiv ist oder nicht.

Timeout Rekeying

Der GCKS führt periodisch ein Rekeying der IKE_SA, sowie der GROUP_SA durch. Dazu muss er sich bei jedem Teilnehmer melden um die SAs zu erneuern. Dabei würde der GCKS jene Teilnehmer erkennen, welche sich ohne Meldung von der Multicast-Gruppe entfernt haben.

Ein Rekeying ist meist nur in grösseren Zeitabständen erforderlich. Aus diesem Grund würde der GCKS einen Dead Peer gegebenenfalls nur sehr spät erkennen können.

Separates Timeout

Damit der GCKS schneller Dead Peers erkennen kann, führt er ein eigenes Timeout pro Peer. Ist dieses abgelaufen, führt er eine Dead Peer Detection durch. Dazu sendet er eine leere Informational-Nachricht. Eine solche Nachricht muss immer bestätigt werden.

Fazit für Austritt ohne Meldung

Die Diskussion mit den Betreuern vom 24.03.2009 hat ergeben, dass ein Timeout beim Rekeying völlig ausreichend ist. Ein zusätzliches Timeout, welches eine Dead Peer Detection DPD zur Folge haben würde (siehe *Separates Timeout*), ist nicht notwendig und wird auch nicht realisiert. Aus der Diskussion hat sich jedoch ein anderer Anwendungsfall einer Dead Peer Detection ergeben (siehe Punkt *3.3.6 Überprüfung GCKS*).

3.3.6 Überprüfung GCKS

Bei verbindungsloser Datenübertragung mittels UPD wird die Datenübertragung nicht garantiert. Erhält ein Teilnehmer für längere Zeit keine Nachrichten mehr über die IKE_SA, so ist er nicht sicher, ob der GCKS netzwerkmässig noch erreicht werden kann.

Damit ein Teilnehmer überprüfen kann, ob der GCKS noch erreicht werden kann und ob er noch ordentlich funktionsfähig ist, kann er ihm eine Informational-Nachricht senden. Der Empfänger einer solchen Nachricht muss diese bestätigen.

Durch diese Möglichkeit der Dead Peer Detection hat der Teilnehmer ein Mittel, die verbindungslose Kommunikation zum GCKS ein wenig verlässlicher zu gestalten.

3.3.7 Nachrichtenaustausch

3.3.7.1 IKE_SA_INIT-Nachricht

Als erstes muss die IKE_SA aufgebaut werden. Da sie im Grunde genommen eine normale IKE_SA gemäss IKEv2-Standard [6] ist, können diese Nachrichtentypen übernommen werden. Mit der IKE_SA_INIT-Nachricht werden die kryptographischen Parameter ausgehandelt, die Nonces ausgetauscht und ein Diffie Hellman Austausch vorgenommen.

Im Header sind jeweils die SPIs (Security Parameter Index) des Initiators (definiert durch den Teilnehmer) wie auch des Responders (definiert durch den GCSK) enthalten. Für SAML-Assertions wird ein Zertifikatstyp definiert, damit die Anforderung im Payload CERTREQ mitgeteilt werden kann. Durch die Notification GROUP_SA_SUPPORTED wird dem Teilnehmer mitgeteilt, dass der Server multicastfähig ist.

Nachrichten

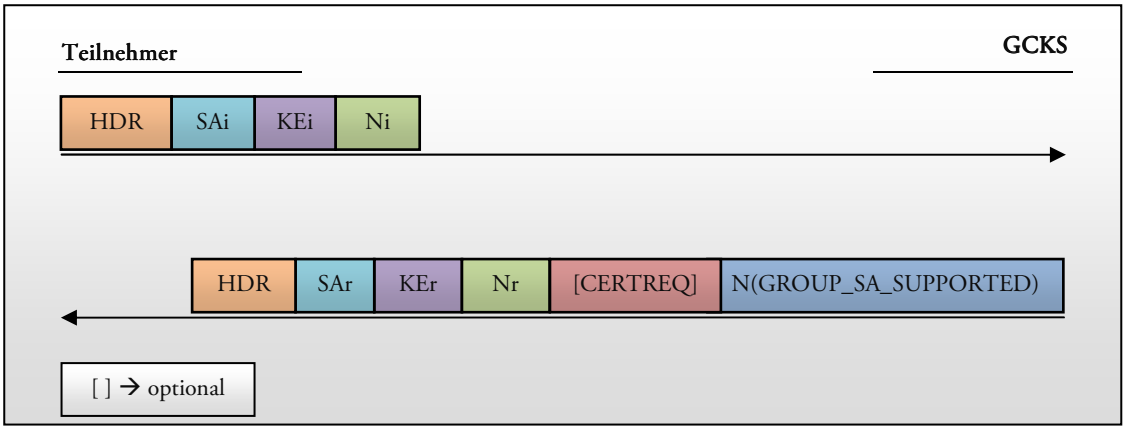

Abbildung 12: Nachrichtenaustausch IKE_SA_INIT

3.3.7.2 IKE_AUTH-Nachricht

Das nächste Nachrichtenpaar (IKE_AUTH) authentisiert die vorherigen Nachrichten, tauscht die Identitäten und allfällige Zertifikate aus und setzt die erste GROUP_SA auf. Dafür werden die Traffic Selectors benötigt. Diese definieren die Adressbereiche der Multicast-Gruppe (Sender- und Multicastgruppenadresse).

Durch die Notification N(GROUP_SA) wird dem GCKS mitgeteilt, dass eine neue GROUP_SA aufgesetzt werden soll. Dies hat zur Folge dass in der Antwort eine Notification GROUP_SA übermittelt wird, welche den TEK enthält.

Nachrichten

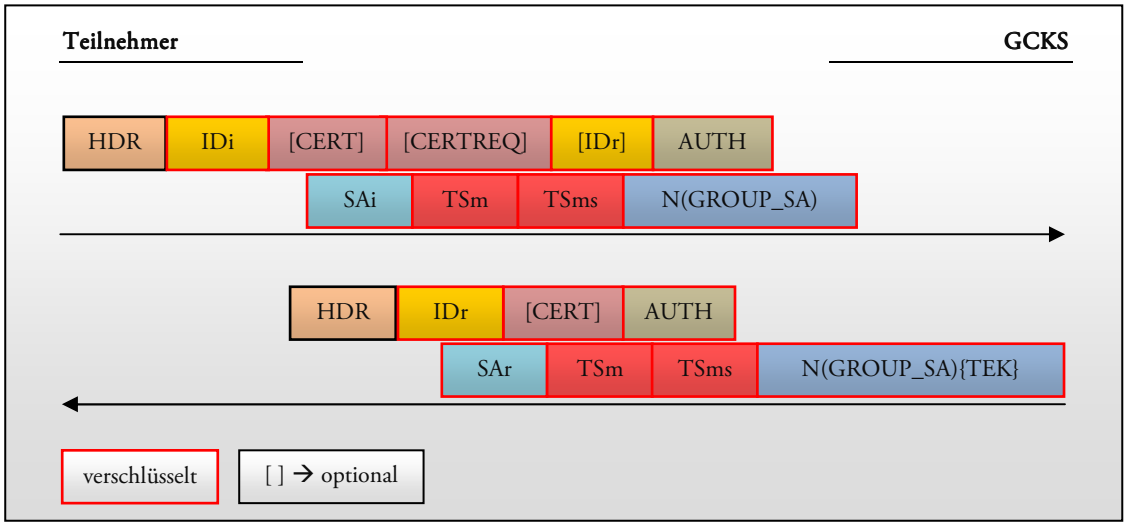

Abbildung 13: Nachrichtenaustausch IKE_AUTH

3.3.7.3 CREATE_CHILD_SA-Nachricht

Beim Rekeying der IKE_SA werden CREATE_CHILD_SA Nachrichten verwendet. Die Informationen können aus dem IKEv2-Standard [6] entnommen werden.

Für den Aufbau weiterer GROUP_SAs kommen ebenfalls CREATE_CHILD_SA-Messages mit den korrekten Notifikationen zum Einsatz. Neue GROUP_SAs werden für Beitritt zu einer Multicastgruppe, das propagieren einer GROUP_SA an einen Teilnehmer, wie auch für das Rekeying einer GROUP_SA benötigt.

Rekeying der IKE_SA

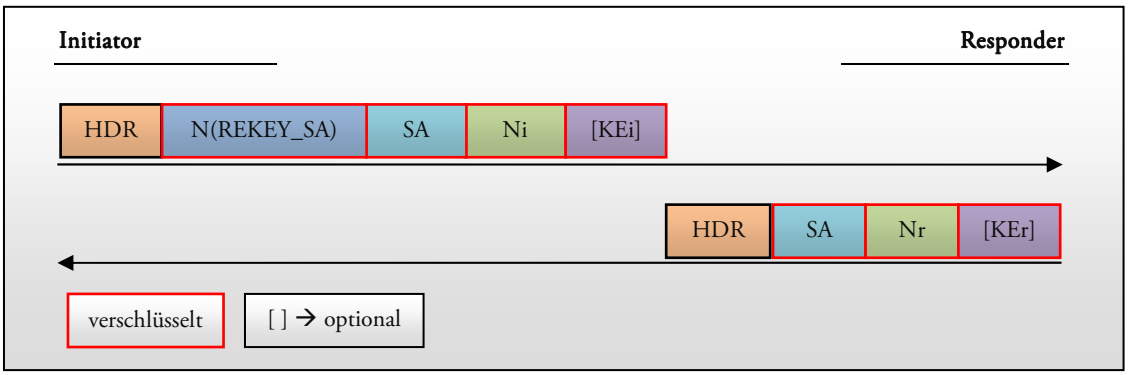

Abbildung 14: Nachrichtenaustausch beim Rekeying der IKE_SA

Damit die Schlüsselinformationen der IKE_SA erneuert werden können, wird mittels CREATE_CHILD_SA-Message eine neue IKE_SA aufgebaut. Die alte IKE_SA kann gelöscht werden, sobald die neue korrekt aufgebaut worden ist. Dabei ist wichtig, dass die neue IKE_SA alle

GROUP_SAs der alten IKE_SA erbt. Es gilt zu beachten, dass die CREATE_CHILD_SA- wie auch die DELETE-Message über die alte IKE_SA versendet bzw. verschlüsselt werden.

Der Aufbau einer Delete-Nachricht kann aus *3.3.7.4 Informational Delete* entnommen werden.

Rekeying einer GROUP_SA

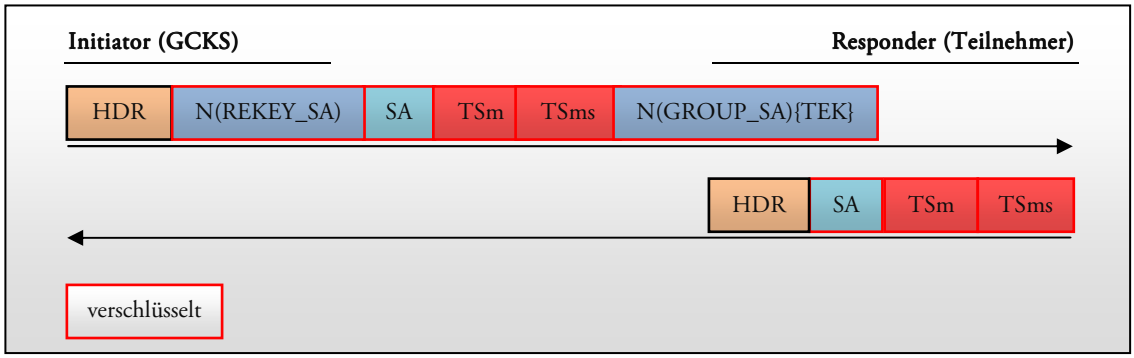

Abbildung 15: Nachrichtenaustausch beim Rekeying der GROUP_SA

Das Grundprinzip für das Rekeying einer GROUP_SA ist das Äquivalent zum Rekeying der IKE_SA. Es wird eine neue GROUP_SA erstellt. Sobald die neue SA aufgebaut wurde, kann die alte gelöscht werden. Da für das Rekeying einer GROUP_SA keine Nonces und Diffie-Hellman-Werte benötigt werden, können diese weggelassen werden.

Zusätzliche GROUP_SA anfordern

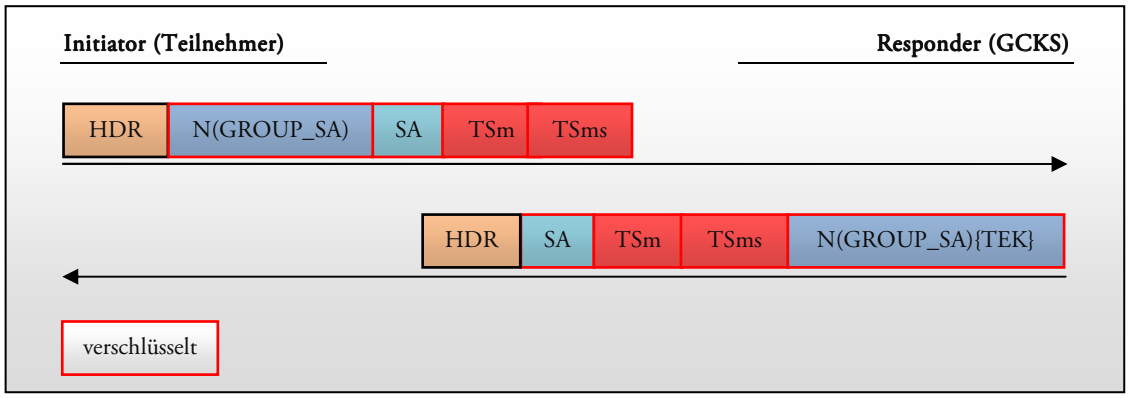

Abbildung 16: Nachrichtenaustausch beim Anfordern einer zusätzlichen GROUP_SA

Wenn ein Teilnehmer eine zusätzliche GROUP_SA anfordert, sendet er eine CREATE_CHILD_SA Nachricht mit einer GROUP_SA Notification. Der GCKS erkennt aufgrund der Traffic Selectors, um welche Gruppe es sich handelt. Er autorisiert den Teilnehmer und sendet den TEK in einem Notify-Payload in der CREATE_CHILD_SA Response zurück.

GROUP_SA an Teilnehmer propagieren

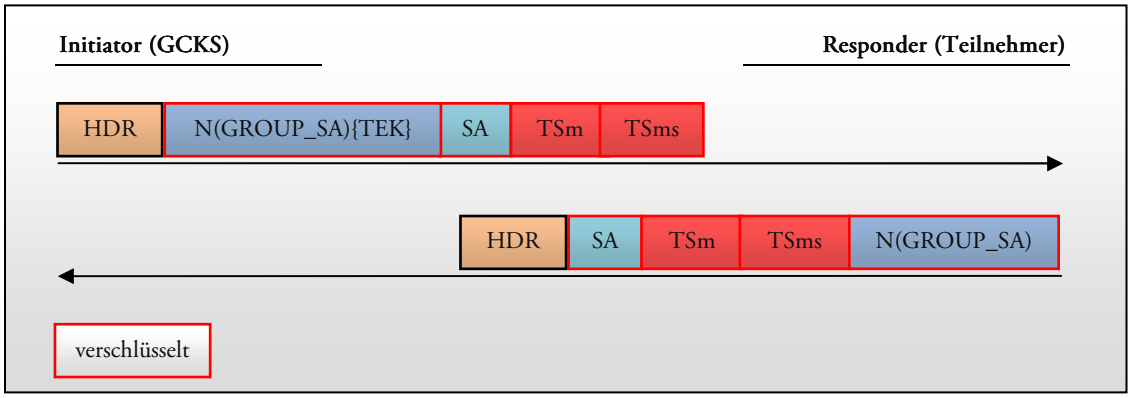

Abbildung 17: Nachrichtenaustausch beim Pushen einer zusätzlichen GROUP_SA

Möchte der GCKS einem Teilnehmer eine GROUP_SA propagieren, sendet er eine CREATE_CHILD_SA-Nachricht mit einer GROUP_SA Notification und dem zu verwendenden Transport Encryption Key TEK. Der Teilnehmer installiert die GROUP_SA mit den erhaltenen Parametern und antwortet darauf mit einer Response.

3.3.7.4 Informational Delete

Das Löschen einer IKE_SA oder GROUP_SA kann vom GCKS oder von einem normalen Teilnehmer ausgelöst werden. Einerseits kann der GCKS das Auflösen einer Multicast-Gruppe mitteilen und andererseits kann ein Teilnehmer seinen Ausstieg aus der Gruppe bekannt geben.

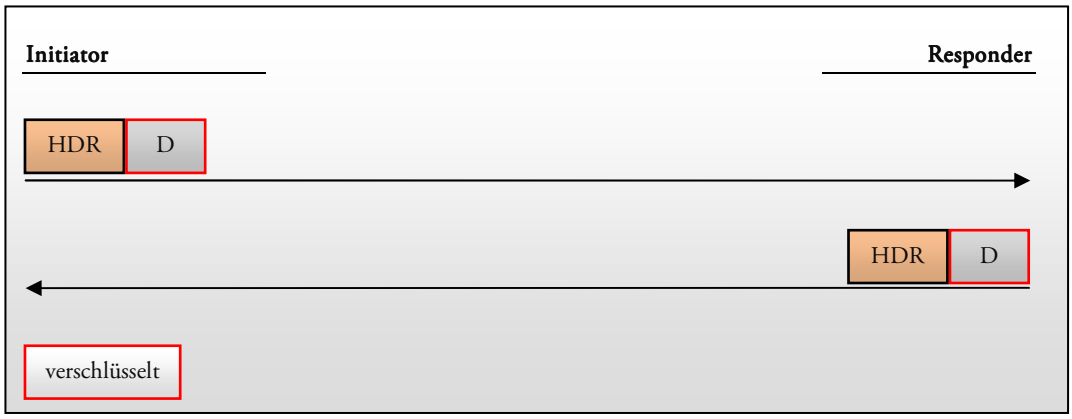

Abbildung 18: Informational-Nachricht zum Abmelden von einer Multicast-Gruppe

Für beide Situationen wird die Informational-Nachricht aus dem IKEv2 Standard [6] verwendet. Dabei wird im Delete-Payload definiert, welche SA(s) gelöscht werden soll(en). Der Responder entfernt die SA und sendet eine Bestätigung mit demselben Inhalt.

3.3.7.5 Dead Peer Detection

Mit einer leeren Informational Nachricht kann ein Teilnehmer herausfinden, ob sein GCKS noch erreichbar ist. Eine solche leere Nachricht muss (wie jede IKEv2 Nachricht) bestätigt werden.

Abbildung 19: Leere Nachricht als Dead Peer Detection

3.3.7.6 Payload

Die verwendeten Payload-Abkürzungen in den IKE-Nachrichten werden in nachfolgender Tabelle kurz erläutert.

Notation	Payload
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
D	Delete
HDR	IKE Header
IDi, IDr	Identification
KE	Key Exchange
Ni, Nr	Nonce
N	Notify
SA	Security Association
TSm	Traffic Selector Multicast
TSms	Traffic Selector Multicast Sender

Tabelle 2: Payload der Nachrichten

Die verschiedenen Notifications sind in folgender Tabelle erklärt.

Notification	Beschreibung
REKEY_SA	Bedeutet, dass die CREATE_CHILD_SA-Nachricht für das Rekeying verwendet wird. Dadurch wird das Rekeying der gewünschten SA (IKE_SA oder GROUP_SA) initiiert.
GROUP_SA	Damit wird die Gegenstelle informiert, dass es sich um eine GROUP_SA handelt. Dies ist bei der IKE_AUTH- und bei der CREATE_CHILD_SA-Nachricht von Bedeutung.
GROUP_SA_SUPPORTED	Mit der GROUP_SA_SUPPORTED-Notification wird dem Teilnehmer mitgeteilt, dass der Responder den Aufbau von GROUP_SAs unterstützt.

Tabelle 3: Notifications für Multicastfunktionalität

3.3.8 Fehlerfälle

In der bisherigen Analysephase wurde hauptsächlich das erfolgreiche Erstellen und Verwalten einer GROUP_SA untersucht. In diesem Abschnitt sollen die verschiedenen Fehlerfälle untersucht werden. Damit sind jene Abläufe gemeint, welche von einem normalen Aufsetzen, Rekeying und Abbauen einer GROUP_SA abweichen.

Das IPsec Multicast Konzept dieses Projekts lehnt sich sehr stark an den IKEv2 Standard an. Aus diesem Grund sind bereits viele Fehlerfälle durch den Standard abgedeckt. Diese werden im Folgenden zwar erwähnt aber nicht detailliert erklärt.

3.3.8.1 IKEv2 Fehler

Folgende Punkte enthalten Fehlerfälle, die bereits durch den IKEv2 Standard abgedeckt werden:

- Geht eine Nachricht verloren, so führt der Sender des Requests eine Retransmission durch.
- Aus den SA Vorschlägen des neuen Teilnehmers kann der GCKS keine geeignete Auswahl treffen. Er antwortet mit der Notify-Meldung NO_PROPOSAL_CHOSEN.
- Erhält ein IPsec Teilnehmer eine Nachricht, die keiner gültigen SPI zugeordnet werden kann, so wird die Nachricht ignoriert. Handelt es sich dabei um ein Request, kann eine Antwort gesendet werden, die einen Notify-Payload enthält, welcher den Fehler beschreibt.
- Falls eine Authentisierung eines Teilnehmers fehlschlägt, so enthält die Antwort ein AUTHENTICATION_FAILED Notify-Payload.

- Falls der GCKS die Traffic Selectors des Initiators nicht zu einem gültigen Bereich einschränken kann, sendet er eine TS_UNACCEPTABLE Notification.

3.3.8.2 Phase 1: IKE_SA_INIT

Der Austausch der IKE_SA_INIT Nachrichten entspricht komplett dem IKEv2 Standard. Aus diesem Grund sind auch die bekannten Fehlerfälle bereits abgedeckt und müssen nicht speziell behandelt werden.

3.3.8.3 Phase 2: IKE_AUTH

In dieser Phase wird die GROUP_SA erstellt. Die folgenden Fehler können dabei auftreten.

Fehler	Falls der GCKS GROUP_SAs unterstützt, teilt er dem neuen Teilnehmer dies in der IKE_SA_INIIT Response-Nachricht mit. Er fügt einen zusätzlichen Notify-Payload ein, der die Information GROUP_SA_SUPPORTED enthält. Falls nun diese Notification fehlt, kann der Teilnehmer nicht mit dem Erstellen einer GROUP_SA weiterfahren.
Lösung	Der Teilnehmer erkennt, dass sein Gegenüber keine GROUP_SAs unterstützt und bricht daher den Multicast-Beitritt ab.

Fehler	Wenn der GCKS den IKE_AUTH Request erhält, muss er unter anderem den Teilnehmer autorisieren. Falls dieser Vorgang fehlschlägt, wird dies dem Client mitgeteilt.
Lösung	Der GCKS fügt in die Response eine Notification (TS_UNACCEPTABLE) ein, die dem Teilnehmer mitteilt, dass die Autorisierung fehlgeschlagen ist. Die IKE_SA wurde jedoch komplett aufgebaut und bleibt bestehen. Der Teilnehmer kann nun versuchen, mit einer CREATE_CHILD_SA Nachricht eine andere GROUP_SA aufzubauen.

3.3.8.4 Phase 3: CREATE_CHILD_SA

Die CREATE_CHILD_SA Nachricht wird entweder für das Erstellen einer zusätzlichen GROUP_SA oder für das Rekeying verwendet. Ein Rekeying kann die IKE_SA oder die GROUP_SA ersetzen. Das Rekeying der IKE_SA entspricht dem IKEv2 Standard und deshalb werden mögliche Fehlerfälle hier nicht weiter beschrieben.

Beim Erstellen einer zusätzlichen GROUP_SA können weitere Fehler auftreten:

Fehler	Auch beim Erstellen einer zusätzlichen GROUP_SA muss der Teilnehmer autorisiert werden. Dieser Vorgang entspricht der Autorisierung während der IKE_AUTH Phase und kann ebenfalls fehlschlagen.
Lösung	Der GCKS teilt dem Sender des Requests mit einer Notification (TS_UNACCEPTABLE) mit, dass er für die gewünschte Gruppe nicht autorisiert ist.

Das Rekeying der GROUP_SA wird jeweils durch den GCKS angestossen. In einer modifizierten CREATE_CHILD_SA Nachricht teilt er die neue SA allen Teilnehmern mit. Dies kann zu Fehlern führen:

Fehler	Der GCKS sendet per Unicast eine CREATE_CHILD_SA Nachricht an alle Teilnehmer. Falls er von einem Teilnehmer keine Bestätigung erhält, meldet ein Timer die fehlende Bestätigung.
Lösung	Nach einer gewissen Anzahl an Wiederholungen wird der Rekeying-Vorgang abgebrochen. Der GCKS entfernt alle SAs, die diesem Teilnehmer zugeordnet sind (IKE_SA und GROUP_SAs). Da das Senden einer DELETE Notification wenig Sinn macht, werden die SAs ohne Mitteilung gelöscht (silent close).

Fehler	Aufgrund eines Netzwerkunterbruchs oder eines anderen Fehlers kann der GCKS dem Teilnehmer nicht mitteilen, dass ein Rekeying einer GROUP_SA vorgenommen wird.
Lösung	Aufgrund einer Dead Peer Detection oder nicht entschlüsselbarem Traffic erkennt der Teilneher, dass er eine ungültige SA besitzt. Er baut darauf die IKE_SA und GROUP_SA ab und meldet sich beim GCKS mit einer IKE_SA_INIT Nachricht neu an.

3.3.9 Multi-Sender-Architektur

Die Erweiterung zu einer Multi-Sender-Architektur hat zur Folge, dass in einer Gruppe mehrere Teilnehmer über die Senderfunktionalität verfügen.

Für den Aufbau einer Infrastruktur mit Multi-Sender-Funktionalität gibt es drei grundsätzlich verschiedene Ansätze.

Szenario 1: Alle Sender dieselbe GROUP_SA

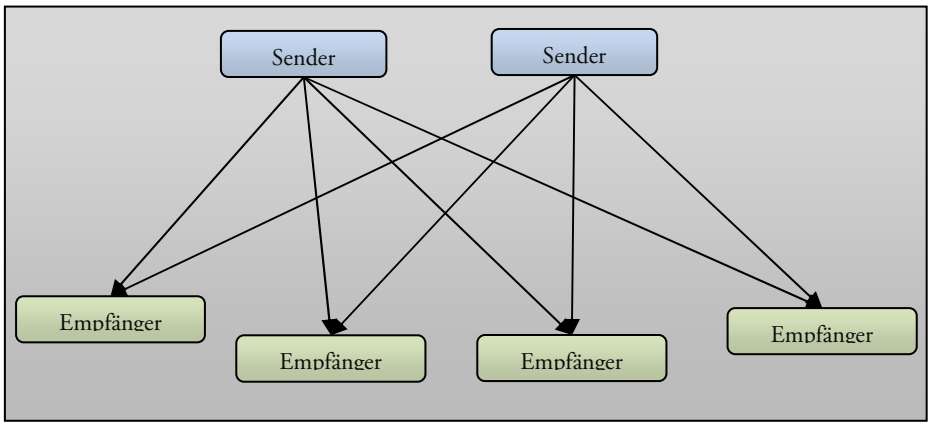

Abbildung 20: Alle Sender verwenden dieselbe GROUP_SA für die Verteilung des Traffics

Das Szenario 1 hantiert mit einer GROUP_SA. Alle Sender verwenden dieselbe Security Association für die Verschlüsselung der Nutzdaten.

Adressierungsvariante 1:

TSm	TS Multicast	Multicast-Adresse	224.1.2.3/32
TSms	TS Multicast Sender	Gruppe von Unicast-Adressen	192.168.1.2/32
			192.168.1.4/32
			192.168.1.25/32
			192.168.1.49/32

Der Traffic Selector des Senders besteht aus mehreren Adressen. Tritt ein neuer Sender einer Gruppe bei, wird die neue IP-Adresse bei einem Rekeying an die anderen Teilnehmer übermittelt. Dadurch ist die Liste aller sendenden Teilnehmer stets aktuell.

Adressierungsvariante 2:

TSm	TS Multicast	Multicast-Adresse	224.1.2.3/32
TSms	TS Multicast Sender	Wildcard-Adresse	0.0.0.0/0

Da jeder Teilnehmer die Pakete mit der gewünschten Absenderadresse versehen kann, ist es für die Sendeberechtigung nicht relevant, welche Adresse im Traffic Selector steht. Aus sicherheitstechnischen Gründen ist die Verwendung einer Wildcard-Adresse im Traffic Selector des Senders mit der Liste aller Sender gleichzusetzen.

Beurteilung des Szenarios

Durch die Verwendung nur einer GROUP_SA für alle Sender ist der Administrationsaufwand für die Teilnehmer minimal. Ein Problem könnte die Sendeautorisierung der Teilnehmer darstellen.

Damit Replay-Attacken verhindert werden können, müssen die sendenden Teilnehmer einer GROUP_SA die Sequenz-Nummer untereinander synchronisieren. Der Aufwand dafür wäre sehr gross und würde ein Performance-Problem darstellen.

Die zwei Adressierungs-Varianten sind bis auf den unterschiedlichen Administrationsaufwand gleichzusetzen.

Szenario 2: Pro Sender eine GROUP_SA

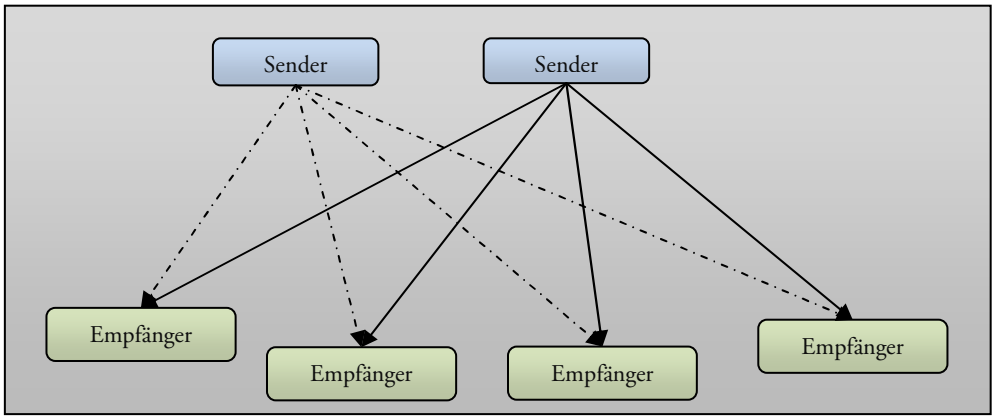

Abbildung 21: Jeder Sender verteilt die Nachrichten über eine eigene GROUP_SA

Das zweite Szenario sieht vor, dass für jeden Sender einer Multicastgruppe eine eigene GROUP_SA erstellt wird. Dabei sieht der Ablauf folgendermassen aus:

- Teilnehmer möchte Daten an eine Multicast-Gruppe Senden. Dabei teilt er dem GCKS in der CREATE_CHILD_SA oder in der IKE_AUTH Nachricht mit, dass er als Sender fungieren möchte.

- Der GCKS erstellt für den neuen Sender eine neue GROUP_SA. Diese wird an alle Teilnehmer propagiert.

Adressierung:

TSm	TS Multicast	Multicast-Adresse	224.1.2.3/32
TSms	TS Multicast Sender	Unicast-Adresse	192.168.1.1/32

Da für jeden Sender eine eigene GROUP_SA aufgebaut wird, befindet sich in jeder GROUP_SA nur ein sendender Teilnehmer. Aus diesem Grund können dieselben Traffic Selectors wie beim Single-Sender-Szenario verwendet werden (siehe *3.3.4.1 Traffic Selectors*). Als Empfängeradresse (*Traffic Selector Multicast*) wird die Multicastadresse der Gruppen angegeben. Diese ist in allen Gruppen der verschiedenen Sender dieselbe. Als Senderadresse (*Traffic Selector Multicast Sender*) wird die Unicastadresse des Senders verwendet.

Beurteilung des Szenarios

Da jeder sendende Teilnehmer innerhalb der Multicast-Gruppe über eine eigene GROUP_SA sendet, können die Sequenz-Nummern wie gehabt zur Verhinderung von Replay-Attacken verwendet werden.

Ebenfalls könnte für die Authentisierung des Absenders eines Multicast-Pakets das TESLA-Verfahren in Betracht gezogen werden.

Bei diesem Szenario könnte die Skalierbarkeit ein Problem darstellen. Wie sich dies in einem Umfeld mit vielen Sendern auswirken würde, bleibt zu überprüfen.

Szenario 3: Verwendung eines Proxys

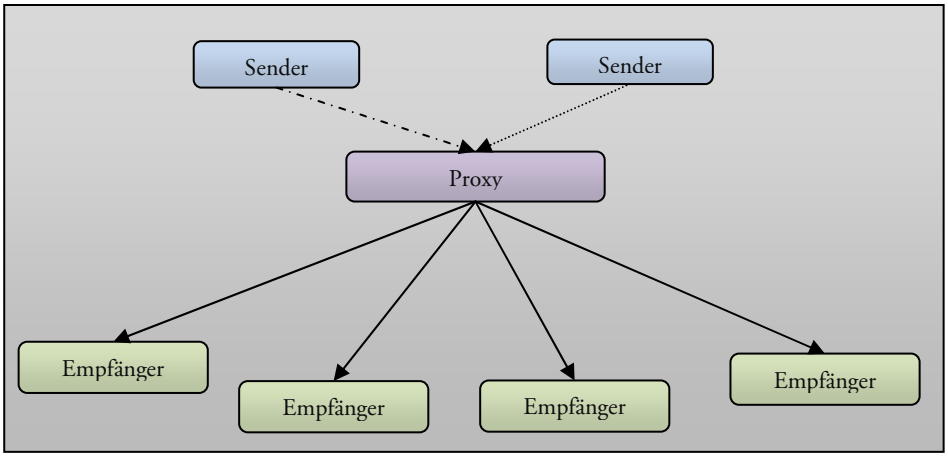

Abbildung 22: Traffic wird über Proxy an die Teilnehmer verteilt

In der dritten Variante wird der gesamte Traffic eines Senders über eine zentrale Stelle geschleust. Dabei hat der GCKS die Funktion eines Sende-Proxys. Jeder sendende Teilnehmer übermittelt seinen Traffic an den GCKS. Dieser verteilt ihn über die GROUP_SAs an die verschiedenen Teilnehmer.

Beurteilung des Szenarios
Die Verwendung eines Proxys ist nicht praktikabel. Bei grossem Verkehrsaufkommen würde dieser als Flaschenhals fungieren. Aus diesem Grund wird dieses Szenario nicht weiter analysiert.

Fazit der Multi-Sender-Architektur

Nach der Gegenüberstellung der verschiedenen Szenarien überwiegen die Vorteile beim Szenario 2. Die saubere Trennung der GROUP_SAs lässt die Verwendung der Sequenznummern zu Verhinderung von Replay-Attacken zu. Die mögliche Implementierung des TESLA-Protokolls und Anlehnung an die Single-Sender-Architektur sprechen ebenfalls für das Szenario 2.

3.3.10 Ablaufszenarien

Im Folgenden werden einige essentielle Ablaufszenarien erläutert. Es werden dabei nur die wichtigsten Aktionen berücksichtigt. Das Aufzeigen aller in Frage kommenden Varianten wäre zu komplex und würde nicht zu einem besseren Verständnis führen.

3.3.10.1 Anmeldung an GCKS

Ausgangslage

Ein GCKS (Group Controller and Key Server) und ein Subscriber sind über ein Netzwerk miteinander verbunden. Der GCKS ist in Betrieb genommen und hat bereits Kontakt zu einigen Multicast-Teilnehmern. Neuen Mitgliedern bietet er die Möglichkeit sich anzumelden. Der abgebildete Subscriber möchte sich nun beim GCKS anmelden und sich für eine Multicast-Gruppe registrieren.

Nachrichten und Aktionen

Zwischen dem Subscriber und dem GCKS besteht noch kein sicherer Kanal. Aus diesem Grunde muss der Subscriber neben der GROUP_SA auch eine IKE_SA zum GCKS aufbauen.

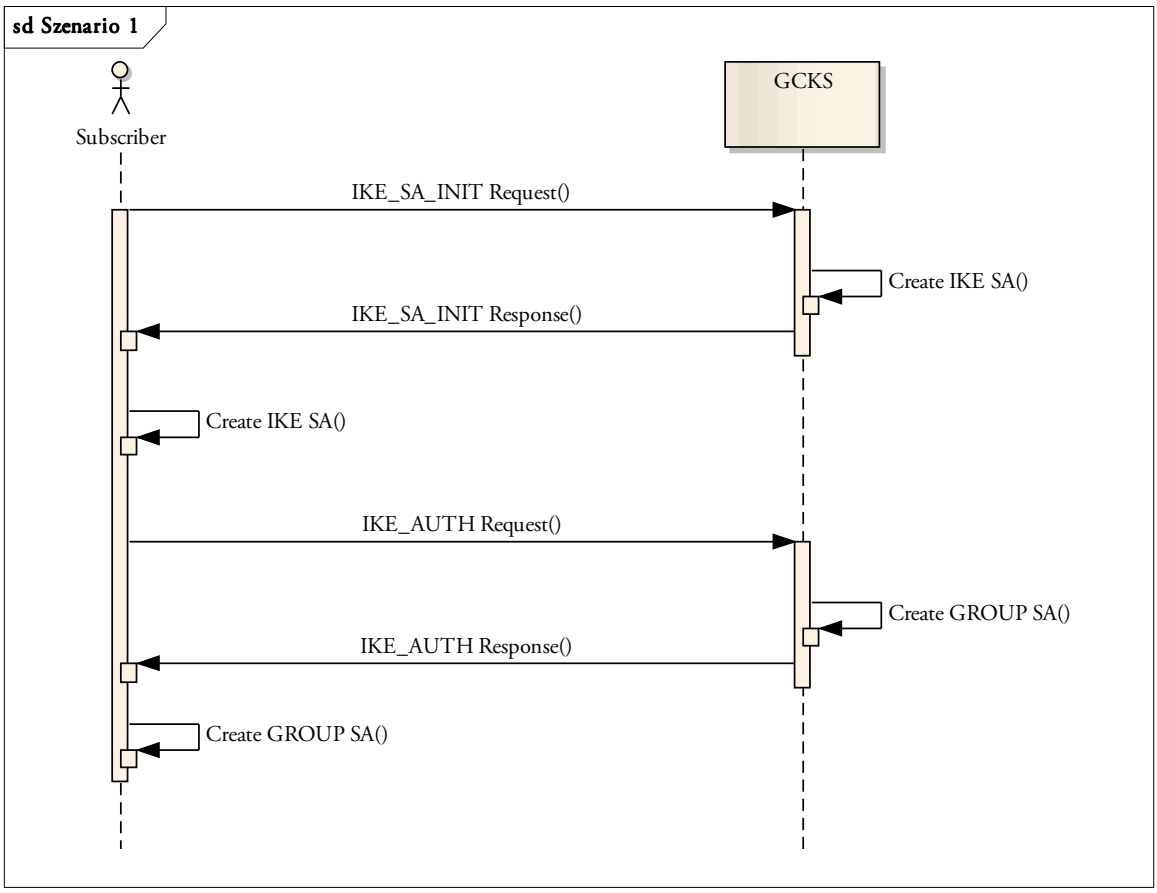

Abbildung 23: Anmeldung an den GCKS als erster empfangender Teilnehmer

Als erstes sendet er dem GCKS einen IKE_SA_INIT Request. Werden die gewünschten Parameter vom GCKS akzeptiert, antwortet er mit einer IKE_SA_INIT Response. Nach Erhalt dieser Antwort hat der Subscriber die IKE_SA zum GCKS aufgebaut.

Im nächsten Schritt wird das IKE_AUTH-Meldungspaar verwendet. Damit kann sich der Benutzer beim GCKS autorisieren und sich für die gewünschte Multicast-Gruppe anmelden. Mittels IKE_AUTH-Request initiiert der Subscriber diesen Vorgang. Wurde der Teilnehmer vom GCKS erfolgreich autorisiert, sendet er dem Teilnehmer das Schlüsselmaterial für die GROUP_SA zurück. Meldet sich der Subscriber als erster empfangender Teilnehmer beim GCKS an, so wird eine neue GROUP_SA erstellt. Falls sich der Subscriber als Sender anmeldet, wird eine neue GROUP_SA erstellt und diese an alle anderen Mitgliedern der Multicast-Gruppe propagiert.

Existieren bereits GROUP_SAs für die gewünschte Multicast-Gruppe, werden diese mittels Rekeying erneuert und an den anmeldenden Subscriber propagiert.

Endzustand

Durch den Verbindungsaufbau mittels IKE_SA_INIT- und IKE_AUTH-Nachrichten wurden eine IKE_SA zum GCKS und mindestens eine GROUP_SA beim Subscriber installiert. Die IKE_SA wird

für die Kommunikation zwischen dem GCKS und dem Teilnehmer verwendet. Rekey-Informationen, Anforderung einer neuen GROUP_SA und ähnliche Befehle können über diesen sicheren Kanal übertragen werden.

Die GROUP_SA wird für das Verschlüsseln oder Entschlüsseln der Nutzdaten des Multicastverkehrs verwendet.

3.3.10.2 Rekeying GROUP_SA

Ausgangslage

Im Netzwerk befinden sich ein GCKS. Dieser verwaltet eine GROUP_SA mit einem sendenden und zwei empfangenden Teilnehmern. Die Lebenszeit der GROUP_SA ist abgelaufen. Mittels Rekeying wird der GCKS das Schlüsselmaterial der GROUP_SA erneuern, um die Sicherheit des Multicast-Verkehrs langfristig zu gewähren.

Nachrichten und Aktionen

Um ein Rekeying durchzuführen muss der GCKS zuerst ein Klon der existierenden GROUP_SA erstellen und dabei das Schlüsselmaterial erneuern. Über das Schlüsselmaterial der neu erstellten GROUP_SA müssen die Multicast-Gruppen-Teilnehmer informiert werden. Damit ein lückenfreier Empfang der verschlüsselten Daten bewerkstelligt werden kann, müssen zuerst alle Empfänger über die neue GROUP_SA informiert werden. Sobald alle den CREATE_CHILD_SA Request mittels Response beantwortet haben, weiss der GCKS, dass alle Teilnehmer die Informationen zur neuen GROUP_SA abgespeichert haben. Danach kann er den Sender mittels CREATE_CHILD_SA Request mit der neuen GROUP_SA vertraut machen. Hat dieser die Schlüsselinformationen erhalten, kann er die Nutzdaten des Multicast-Verkehrs damit verschlüsseln und an die Teilnehmer verteilen. Wurde dem GCKS dies mit einer Response mitgeteilt, sind alle Teilnehmer auf dem neusten Stand. Abschliessend werden die Aufräumarbeiten erledigt. Da die alte GROUP_SA keine Verwendung mehr findet, kann der GCKS alle Teilnehmer dazu auffordern, die alte SA zu löschen. Haben dies alle mit einer Response bestätigt, löscht auch der GCKS die alte GROUP_SA aus seiner Datenbank.

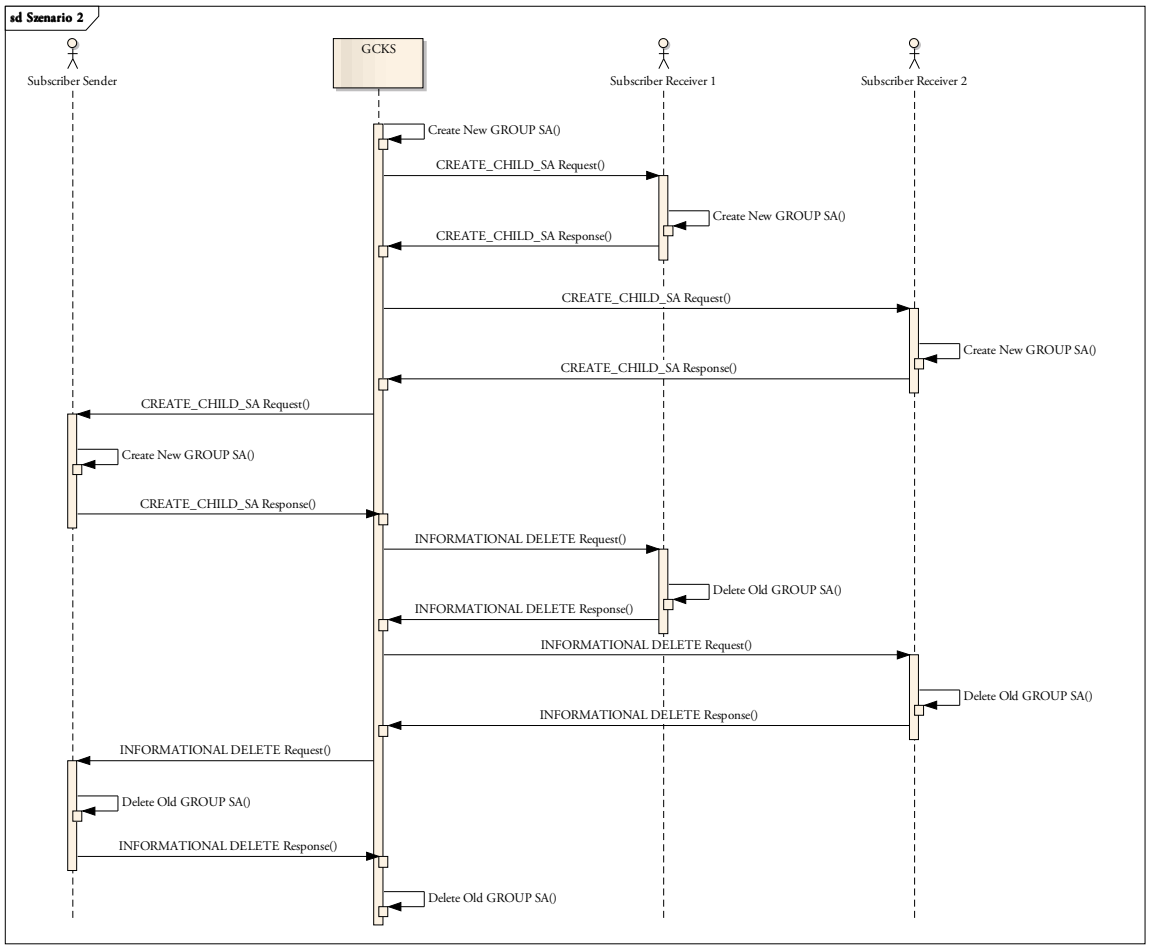

Abbildung 24: Rekeying einer GROUP_SA

Endzustand

Alle Teilnehmer sind mit dem Schlüsselmaterial der neuen GROUP_SA ausgestattet. Der Sender verschlüsselt damit den Multicast-Verkehr. Die alte GROUP_SA findet keine Verwendung mehr und wurde bei allen Teilnehmern entfernt.

3.3.10.3 Neuer Sender

Ausgangslage

Mit dem GCKS sind bereits drei Teilnehmer über eine IKE_SA verbunden. Während der Sender 1 und Receiver 1 in einer gemeinsamen Multicast-Gruppe sind, befindet sich der Sender 2 in einer anderen.

Der Sender 2 möchte ebenfalls Daten senden. Dazu muss eine neue GROUP_SA erstellt werden und an die anderen zwei Subscriber (Sender 1 und Receiver 1) propagiert werden.

Nachrichten und Aktionen

Der Sender 2 meldet sich beim GCKS und fordert mittels CREATE_CHILD_SA Request eine neue GROUP_SA an, in der er der sendende Teilnehmer sein soll. Der GCKS erstellt darauf die neue GROUP_SA. Bevor er dies dem Sender 2 jedoch bestätigen kann, muss er alle bestehenden Multicast-Teilnehmer über die neue GROUP_SA informieren. Mittels CREATE_CHILD_SA-Request an den Sender 1 und Receiver 1 fordert er diese dazu auf die GROUP_SA zu installieren. Hat der GCKS die Responses erhalten, ist die neue GROUP_SA auf allen bestehenden Teilnehmer installiert. Um eine hohe Sicherheit zu gewähren, muss auf allen bestehenden GROUP_SAs der Multicast-Gruppe ein Rekeying durchgeführt werden. Wurde dies abgeschlossen, kann die Response an den Sender 2 zurückgesendet werden. Abschliessend werden die bestehenden GROUP_SAs an den Sender 2 propagiert.

Alle Teilnehmer sind darauf mit dem neusten Schlüsselmaterial aller GROUP_SAs der Multicast-Gruppe ausgestattet.

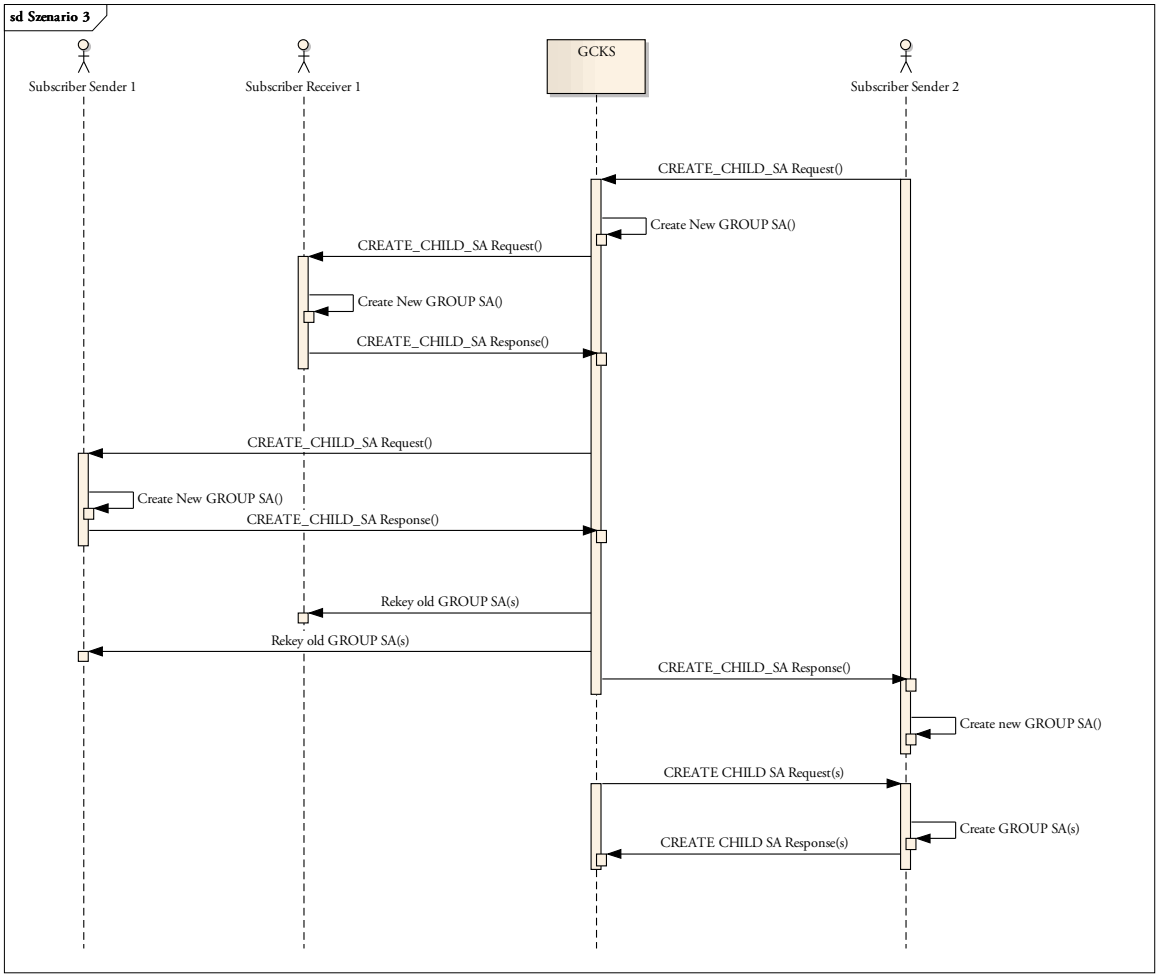

Abbildung 25: Neuer Sender

Endzustand

Der neue Sender 2 wie auch die bereits existierenden Multicast-Gruppenmitglieder Sender 1 und Receiver 1 sind im Besitz der neuen GROUP_SA. Sender 2 kann verschlüsselte Multicast-Pakete an die anderen Teilnehmer übermitteln. Ebenfalls hat er die Möglichkeit die Daten, welche vom Sender 1 versendet werden, zu entschlüsseln.

3.3.10.4 Abmelden empfangender Teilnehmer

Ausgangslage

Der GCKS verwaltet eine GROUP_SA, welche vom Sender 1 und Receiver 1 & 2 benutzt wird. Der empfangende Teilnehmer Receiver 1 benötigt den Multicast-Verkehr von Sender 1 nicht mehr und möchte aus der Multicast-Gruppe austreten.

Nachrichten und Aktionen

Der Austritt aus der GROUP_SA kann durch den Teilnehmer mittels Informational Delete Nachricht initiiert werden. Mit einem Request teilt der Teilnehmer dem GCKS mit, dass er die GROUP_SA nicht mehr benötigt. Da es noch weitere Teilnehmer der GROUP_SA gibt, muss das Schlüsselmaterial mittels Rekeying erneuert werden. Damit wird unterbunden, dass der Receiver 1 den Multicast-Verkehr weiterhin entschlüsseln kann. Für eine genaue Ausführung des Rekeying sei auf *3.3.10.2 Rekeying GROUP_SA* verwiesen.

Nach Abschluss des Rekeying kann der GCKS den Request des Receiver 1 mittels Response bestätigen. Sobald dieser beim Receiver 1 eingetroffen ist, kann er die alten Schlüsselinformationen der GROUP_SA entfernen.

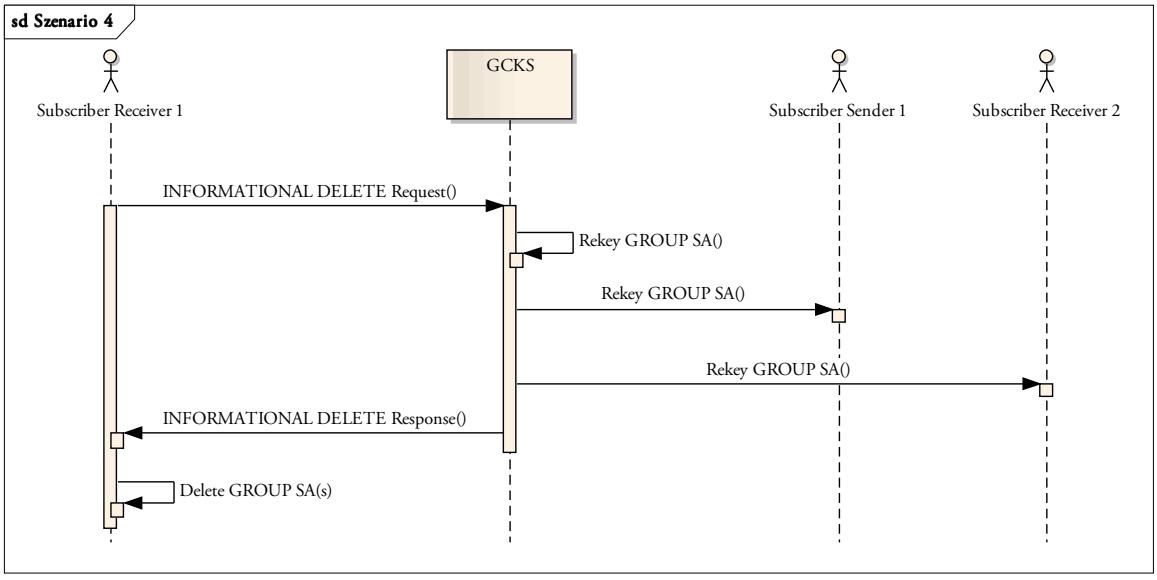

Abbildung 26: Abmelden empfangender Teilnehmer

Endzustand

Durch das Rekeying der GROUP_SA hat der Receiver 1 keine Möglichkeit mehr, den Multicast-Verkehr vom Sender 1 zu entschlüsseln. Die GROUP_SA wurde aus der Datenbank vom Receiver 1 gelöscht.

3.3.10.5 Abmelden sendender Teilnehmer

Ausgangslage

Für die GROUP_SA vom Sender 1 haben sich zwei Receiver registriert. Der Sender möchte aus der Multicast-Gruppe austreten, da er keine weiteren Daten versenden oder empfangen muss.

Nachrichten und Aktionen

Der Sender 1 teilt dem GCKS mittels Informational Delete Request mit, dass er die Multicast-Gruppe verlassen möchte. Durch das Austreten des Senders aus einer GROUP_SA wird diese unbrauchbar und kann auf allen anderen Teilnehmern ebenfalls gelöscht werden. Mit Hilfe von Informational Delete Nachrichtenpaaren wird diese Aktion ausgeführt.

In den anderen GROUP_SAs war der Sender 1 als empfangender Teilnehmer registriert. Aus diesem Grund muss das Schlüsselmaterial dieser GROUP_SAs durch ein Rekeying erneuert werden.

Abschliessend kann der Sender 1 über den erfolgreichen Löschvorgang der GROUP_SA mittels Response informiert werden. Dieser löscht die GROUP_SA und stellt das Versenden des Multicast-Verkehrs ein.

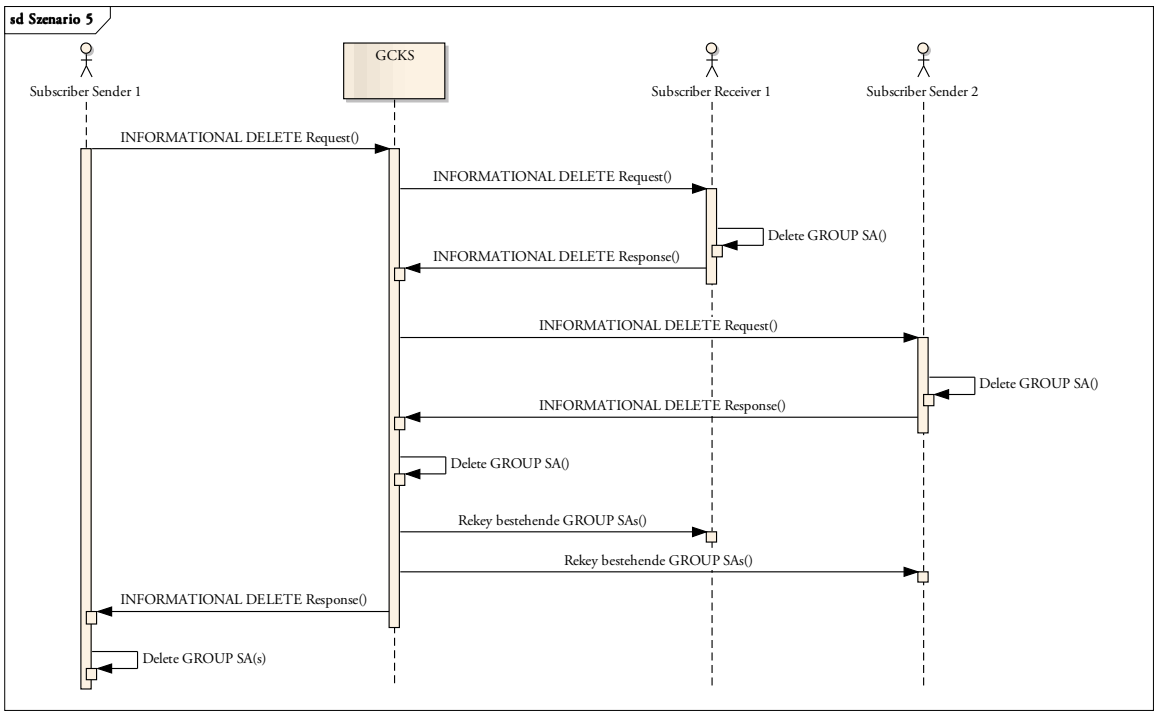

Abbildung 27: Abmelden eines sendenden Teilnehmers

Endzustand

Der sendende Teilnehmer wurde aus der Multicast-Gruppe entfernt. Die nicht weiter benötigte GROUP_SA wurde auf den anderen Teilnehmern gelöscht, während alle anderen GROUP_SAs der Multicast-Gruppe mittels Rekeying erneuert wurden. Die Kommunikation zwischen den verbleibenden Teilnehmer bleibt bestehen.

4 Design

Damit die Stärken und Schwächen des erarbeiteten Konzepts besser zum Vorschein kommen, wurde das Protokoll in Form eines Proof-of-Concept implementiert. Im Folgenden wird der Aufbau dieses fIPsec Multicast Simulators genauer erklärt.

4.1 Entwicklungsebene

Der Simulator wurde komplett auf Applikationsebene entwickelt. Es wurde also nicht auf der Implementierung von strongSwan aufgebaut, sondern eine neue Applikation in C# geschrieben, welche das Protokoll simuliert.

Ein Grund für die Neuentwicklung war vor allem die lange Einarbeitungszeit, welche notwendig gewesen wäre, falls das Projektteam das Protokoll direkt in strongSwan integriert hätte. Ebenfalls waren nur minimale Kenntnisse in der Programmiersprache C vorhanden, was die genannte Einarbeitungszeit noch zusätzlich verlängert hätte. Zudem war auch unklar, ob sich Multicast-Security-Associations im Linux-Kernel überhaupt installieren lassen würden.

4.2 Architekturübersicht

Bevor die erstellten Pakete und Klassen genauer erklärt werden, soll eine grobe Übersicht einen Überblick verschaffen. Es wird aufgezeigt welche Komponenten beim Versenden, respektive Empfangen von Multicast-Daten verwendet werden.

Die Abbildung 28 zeigt die involvierten Objekte beim Versenden von Daten. Das *ProgramSubscriber* stellt hier die Applikation dar, welche über den *IpSecMulticastService* gesicherte Multicast-Daten versendet. Der *IpSecMulticastService* leitet die Nachricht an den entsprechenden *MulticastService* weiter, welcher über den *SaManager* eine GROUP_SA anfordert, mit der er die Daten verschlüsseln kann. Nach dem das Paket verschlüsselt wurde, sendet der *MulticastService* das Paket über einen UDP-Socket an die entsprechende Multicast IP-Adresse.

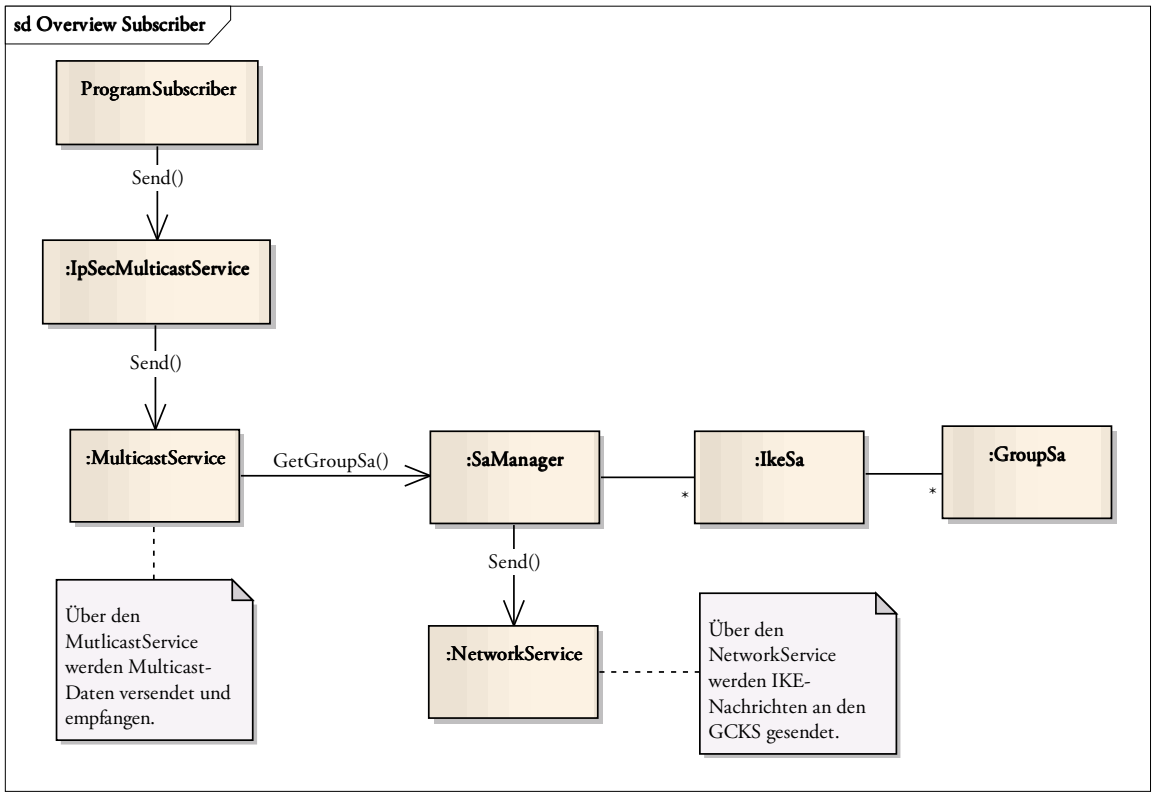

Abbildung 28: Übersicht des Aufbaus eines IPsec Multicast Subscribers

Falls der *SaManager* keine passende GROUP_SA liefern kann, muss er beim GCKS eine solche anfordern. In diesem Fall gehen wir davon aus, dass kein lokaler GCKS vorhanden ist und der *SaManager* sich über das Netzwerk mit dem GCKS in Verbindung setzen muss. Dazu werden die entsprechenden IKE-Nachrichten erstellt und über den *NetworkService* versendet. Nachdem die GROUP_SA erfolgreich angefordert wurde, kann diese wie gehabt dem *MulticastService* zurückgegeben werden.

Der GCKS ist der zentrale Dienstleister in der entwickelten IPsec Multicast Umgebung. Er nimmt Anfragen von Subscribern entgegen und verteilt entsprechend die GROUP_SAs. Die Abbildung 29 zeigt die verwendeten Komponenten. Der GCKS besteht aus einem erweiterten *SaManager*. Die Funktionalität für das Verwalten von IKE_SAs, bzw. GROUP_SAs konnte vom „normalen“ Subscriber wiederverwendet werden.

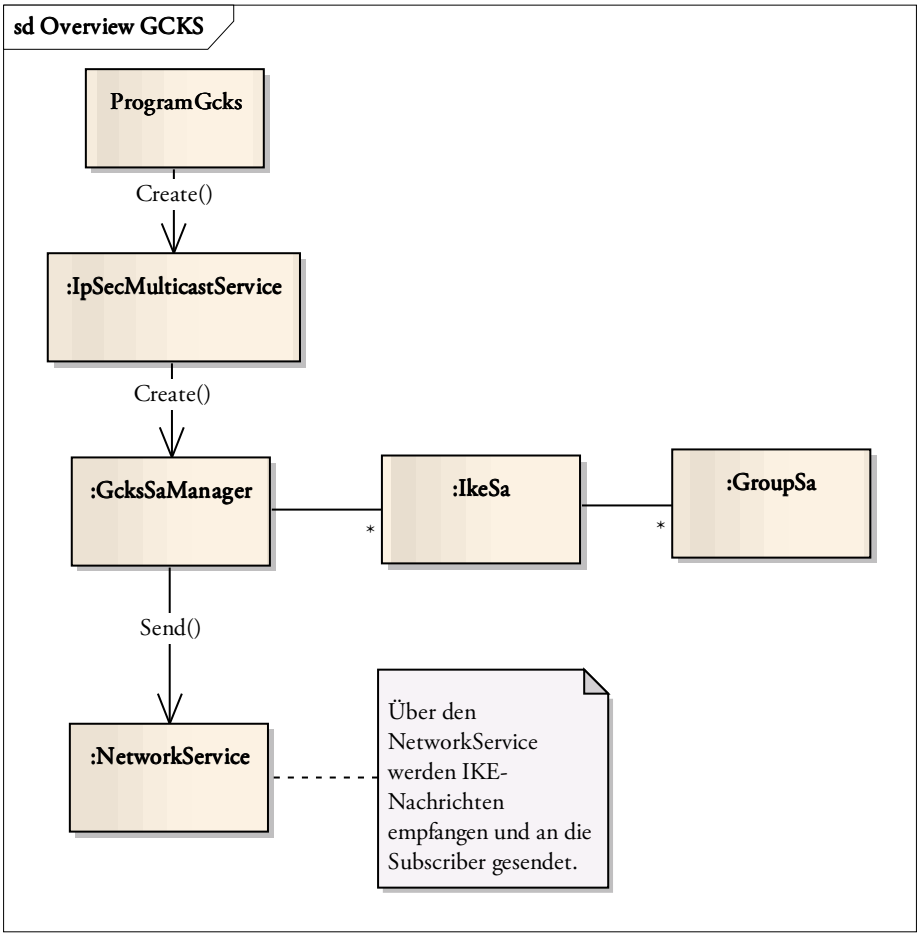

Abbildung 29: Übersicht des Aufbaus des GCKS

Der *GcksSaManager* erweitert den *SaManager* um die Funktion neue GROUP_SAs zu erstellen und an die entsprechenden Subscriber zu verteilen. Um dies durchführen zu können, ist der *GcksSaManager* fähig, weitere IKE-Nachrichten zu verarbeiten. Die Details zu den verschiedenen Komponenten werden in den nachfolgenden Abschnitten erklärt.

4.3 Assemblies

Die erstellte .NET Solution enthält sechs Assemblies. Diese Aufteilung ergab sich aus Strukturgründen und damit das Projekt sauber und einfach getestet werden konnte. Vorerst soll die Tabelle 4 nur eine kurze Übersicht geben, welche Assemblies erstellt wurden. Die genaue Funktionsweise jedes Assemblies und deren Konfiguration sind im Abschnitt *5.4 Anleitung Installation und Konfiguration* beschrieben.

Assembly	Erklärung
IpSecMulticast.dll	Dies ist die Hauptbibliothek, welche die Funktionalität für eine IPsec Multicast Subscriber oder GCKS bereitstellt. Alle folgenden Assemblies verwenden diese Bibliothek für das Ausführen ihrer Aufgaben.
GcksApplication.exe	Das Program GcksApplication.exe startet einen GCKS, der von anderen Subscribern im Netzwerk zur Gruppenverwaltung verwendet werden kann.
ReceiverApplication.exe	Diese Applikation nimmt Verbindung mit dem konfigurierten GCKS auf, fordert eine GROUP_SA an und empfängt fortan alle Daten, welche auf dieser IPsec Multicast Gruppe gesendet werden.
SenderApplication.exe	Beim Start der SenderApplication.exe wird ebenfalls eine Verbindung zum GCKS aufgebaut und eine GROUP_SA angefordert. Im Abstand von 3 Sekunden wird anschliessend fortlaufend eine Nachricht über die angeforderte GROUP_SA gesendet.
Simulator.exe	Dieses Program vereint die drei Applikationen GCKS, Receiver und Sender. Mit Programmargumenten und Konsolenbefehlen wird der Simulator gesteuert und kann so die verschiedenen Funktionen der *IpSecMulticast*-Bibliothek ausführen.
Test.exe	In diesem Assembly sind die Test-Szenarios enthalten. Die Tests können entweder direkt im Visual Studio ablaufen oder es kann pro Peer eine eigene Konsole gestartet werden, damit die Log-Ausgaben überprüft und unterschieden werden können.

Tabelle 4: Assemblies in der .NET Solution

Der Grund für das Erstellen eines GCKS, Receivers und Senders liegt im einfacheren Testen während den Programmierarbeiten. Die drei Applikationen können gleichzeigt vom Visual Studio aus im Debug-Modus gestartet werden. Dadurch kann die Funktionalität des erstellten Programmcodes schnell verifiziert werden und in den einzelnen Konsolenfenstern pro Applikation kann der Ablauf genau verfolgt werden.

4.4 Namespaces

Der Default Namespace der erstellten Software ist *IpSecMulticast*. Dieser Namespace wurde unterteilt in die Namensräume *Network*, *Protocol* und *SecurityAssociation*. Die Tabelle 5 erklärt die Bedeutung der verschiedenen Namespaces.

Namespace	Erklärung
IpSecMulticast	*IpSecMulticast* ist der Root-Namespace. In ihm ist die einzige öffentliche Klasse enthalten, welche als Schnittstelle für die Verwendung der erstellten IPsec Multicast Bibliothek benötigt wird.
IpSecMulticast.Network	Die Funktionalität für das Versenden von Daten über das Netzwerk ist in diesem Namensraum enthalten. Dazu gehört das Versenden und Empfangen von IKE-Nachrichten, wie auch der effektive Datenaustausch über IP Multicast.
IpSecMulticast.Protocol	Alle benötigten Klassen für das Zusammenstellen von IKE-Nachrichten sind in diesem Namespace. Ebenfalls ist eine Klasse vorhanden, welche die verschiedenen Nachrichten-Typen erstellen kann.
IpSecMulticast.SecurityAssociation	Der Namespace *SecurityAssociation* enthält die Klassen für die Verwaltung der Security Associations. Zudem ist hier Funktionalität enthalten, die überprüft, ob ein gewisser Message-Typ zu einem gegebenen Zeitpunkt empfangen werden darf oder nicht.
Program	Die Klassen für die verschiedenen Konsolenapplikationen zum Ausführen und Simulieren verschiedener Szenarien sind im Namespace *Program* enthalten.
Test	Der Namensraum *Test* enthält die Funktionalität für das automatische Testen im Stile von Unit-Tests.

Tabelle 5: Erklärung der erstellten Namespaces

Bevor auf die verschiedenen Namespaces im Detail eingegangen wird, soll eine Übersicht über alle Namespaces das Zusammenspiel verdeutlichen. Die Abbildung 30 zeigt nochmals alle Namensräume mit den wichtigsten darin enthaltenen Klassen und zeigt deren Assoziationen.

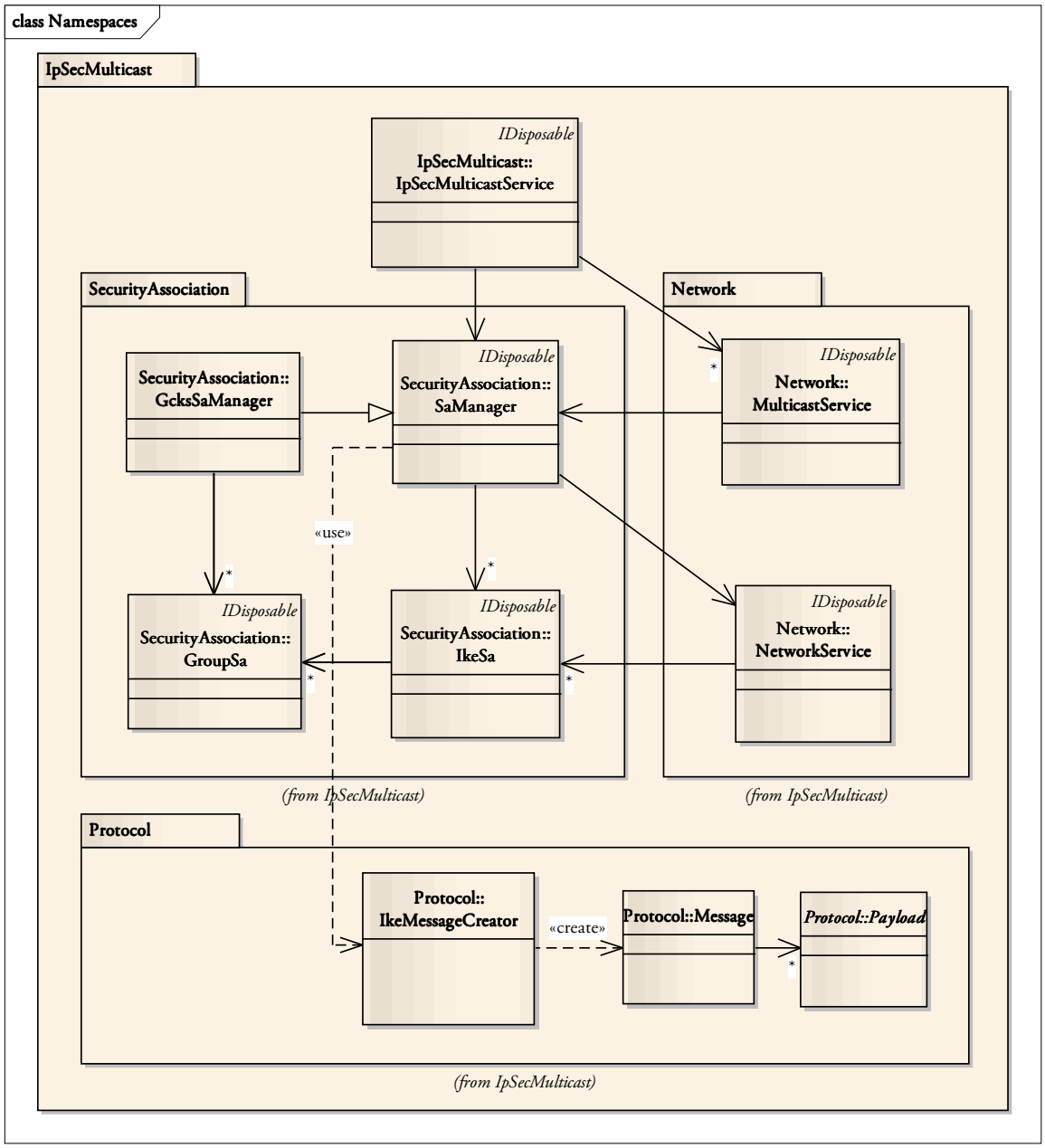

Abbildung 30: Zusammenspiel der Namespaces

In *SecurityAssociation* werden die IKE_SAs und GROUP_SAs abgebildet und verwaltet. Über die in *IpSecMulticast.Netwok* enthaltenen Klassen werden IKE-Nachrichten und Multicast-Daten versendet und empfangen. Der Namensraum *Protocol* enthält die Klassen für das Zusammenstellen von IKE-Nachrichten.

4.4.1 Namespace IpSecMulticast

Die einzige Klasse, welche in diesem Namespace vorhanden ist, heisst *IpSecMulticastService*. Diese Klasse bildet die Schnittstelle zu einer Applikation, welche die erstellte IPsec Multicast Bibliothek nutzen möchte.

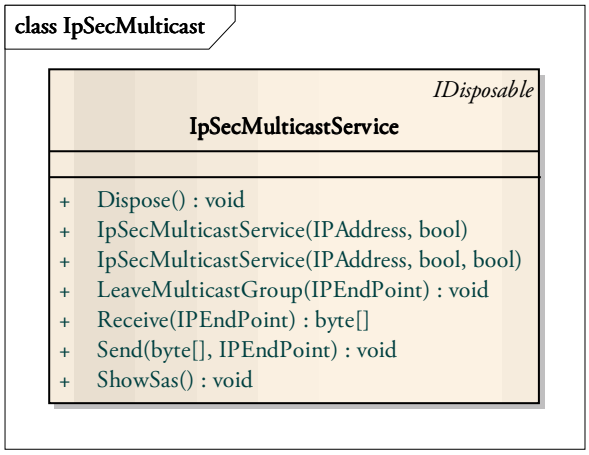

Abbildung 31: Klassen im Namespace IpSecMulticast

Die Abbildung 31 zeigt die erwähnte Klasse mit den Grundoperationen für die Verwendung der *IpSecMulticast*-Bibliothek. Die Methoden der Klasse werden in der Tabelle 6 beschrieben.

Methode	Erklärung
Receive()	Diese Operation führt dazu, dass auf dem angegebenen Multicast IP Endpunkt auf eine Nachricht gewartet wird und diese mit der entsprechenden GROUP_SA entschlüsselt wird.
Send()	*Send()* ist das Gegenstück zu *Receive()* und sendet verschlüsselte Multicast Daten an den übergebenen IP Endpunkt.
LeaveMulticastGroup()	Das Ausführen dieser Operation führt dazu, dass eine Multicast-Gruppe verlassen wird. Dazu gehört das Abmelden beim GCKS und das Verlassen der Gruppe mit der entsprechenden IGMP Nachricht.
ShowSas()	Wenn diese Methode aufgerufen wird, gibt der *IpSecMulticastService* alle aktuell vorhandenen IKE_SAs und die dazugehörigen GROUP_SAs auf der Konsole aus.

Tabelle 6: Methoden der Klasse IpSecMulticastService

4.4.2 Namespace IpSecMulticast.Network

Das Klassendiagramm in der Abbildung 32 enthält die Klassen *MulticastService* und *NetworkService*. Der *MulticastService* versendet und empfängt Multicast-Nachrichten, währenddessen der *NetworkService* IKE-Nachrichten per Unicast übertragen und empfangen kann.

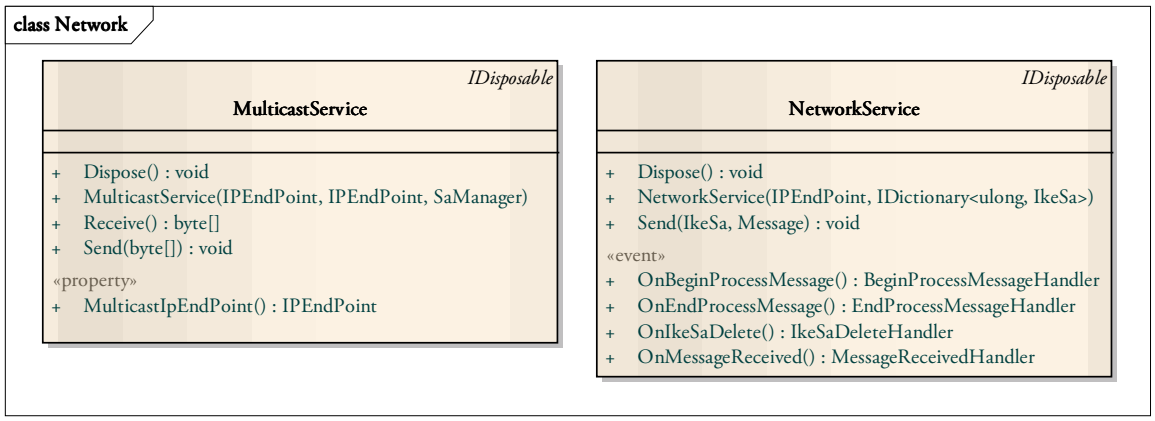

Abbildung 32: Klassen im Namespace IpSecMulticast.Network

4.4.3 Namespace IpSecMulticast.SecurityAssociation

Die Logik für die Verwaltung der verschiedenen Security Associations ist im Namespace *IpSecMulticast.SecurityAssociation* enthalten. Die Klasse *SaManager* verwaltet die IKE_SAs sowie die dazugehörigen GROUP_SAs und führt dies sowohl für einen „normalen“ Subscriber als auch für den GCKS durch.

Die abgeleitete Klasse *GcksSaManager* ist für das Erstellen, das Verteilen und das Rekeying von GROUP_SAs verantwortlich. Die Funktionsweise wird im Abschnitt *4.5 Designentscheide* genauer erläutert.

Die Abbildung 33 zeigt auch die Definitionen der beiden Security Association Typen IKE_SA und GROUP_SA. Die beiden entsprechenden Klassen bestehen hauptsächlich aus Feldern und Properties. Für die Klasse *IkeSa* wird ein Status geführt. Dieser wird für die Überprüfung einer eintreffenden Nachricht verwendet (siehe *4.5.2 State Pattern IkeSa*).

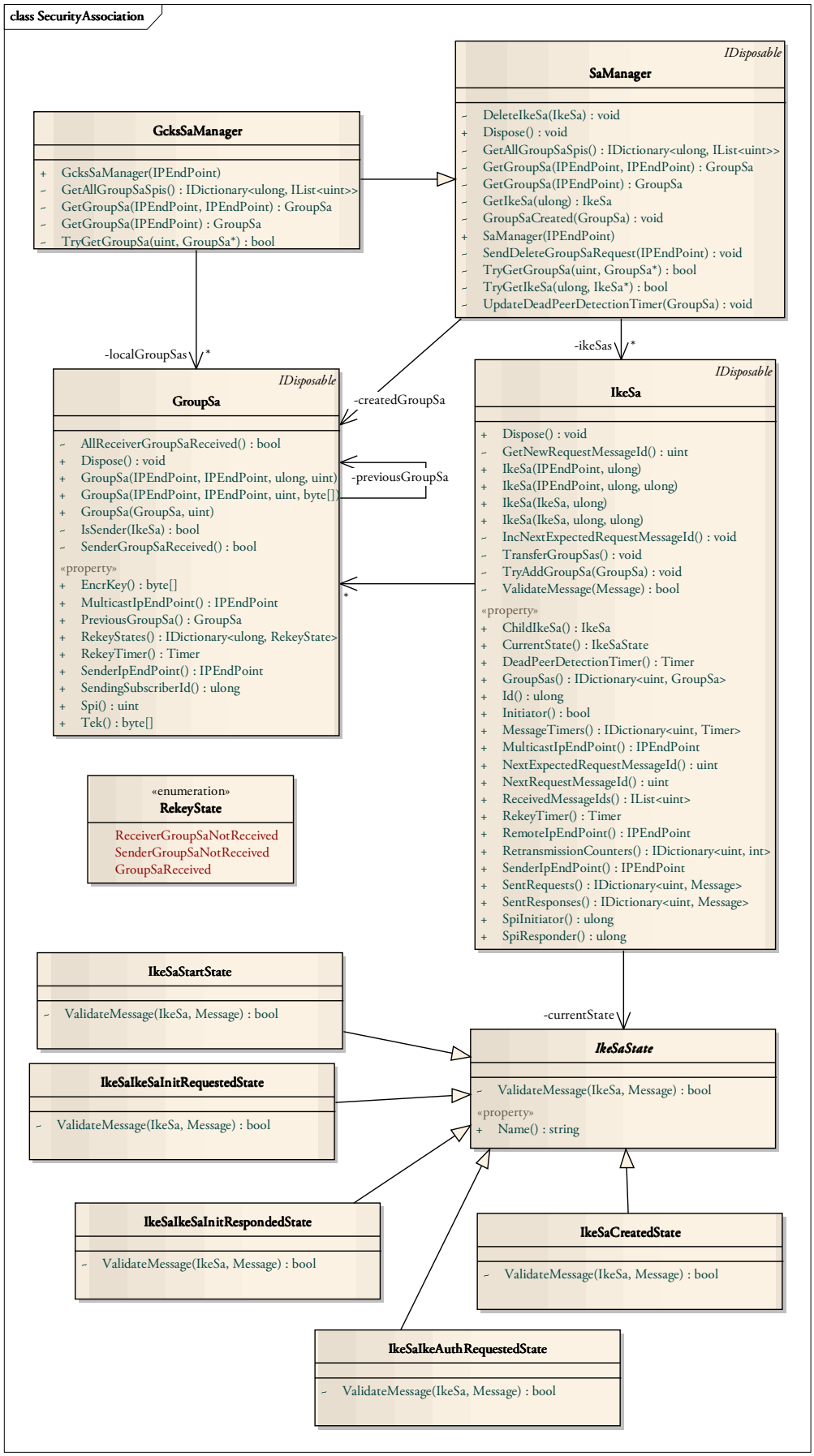

Abbildung 33: Klassen im Namespace IpSecMulticast.SecurityAssociation

4.4.4 Namespace IpSecMulticast.Protocol

Zur besseren Übersicht wurde der Namespace *IpSecMulticast.Protocol* auf zwei Abbildungen aufgeteilt. Im ersten Klassendiagramm in der Abbildung 34 ist ersichtlich, wie eine IKE-Nachricht erstellt wird.

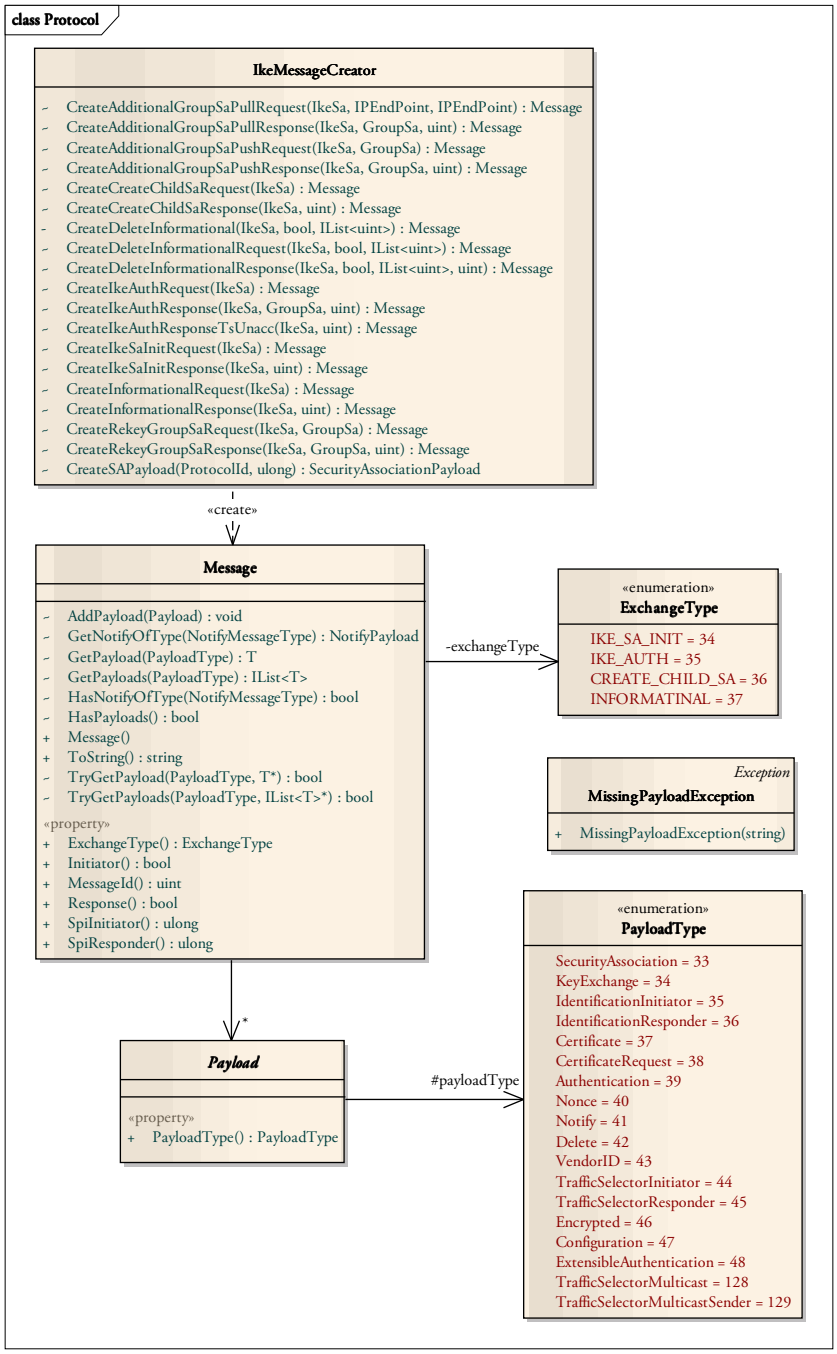

Abbildung 34: Klassen im Namespace IpSecMulticast.Protocol (1)

Jede Methode der Factory Klasse *IkeMessageCreator* erstellt eine IKE-Nachricht mit einem bestimmten Nachrichten *ExchangeType*. Eine solche *Message* enthält in der Regel mehrere *Payloads*, welche aufgrund ihres *PayloadTypes* richtig interpretiert werden können.

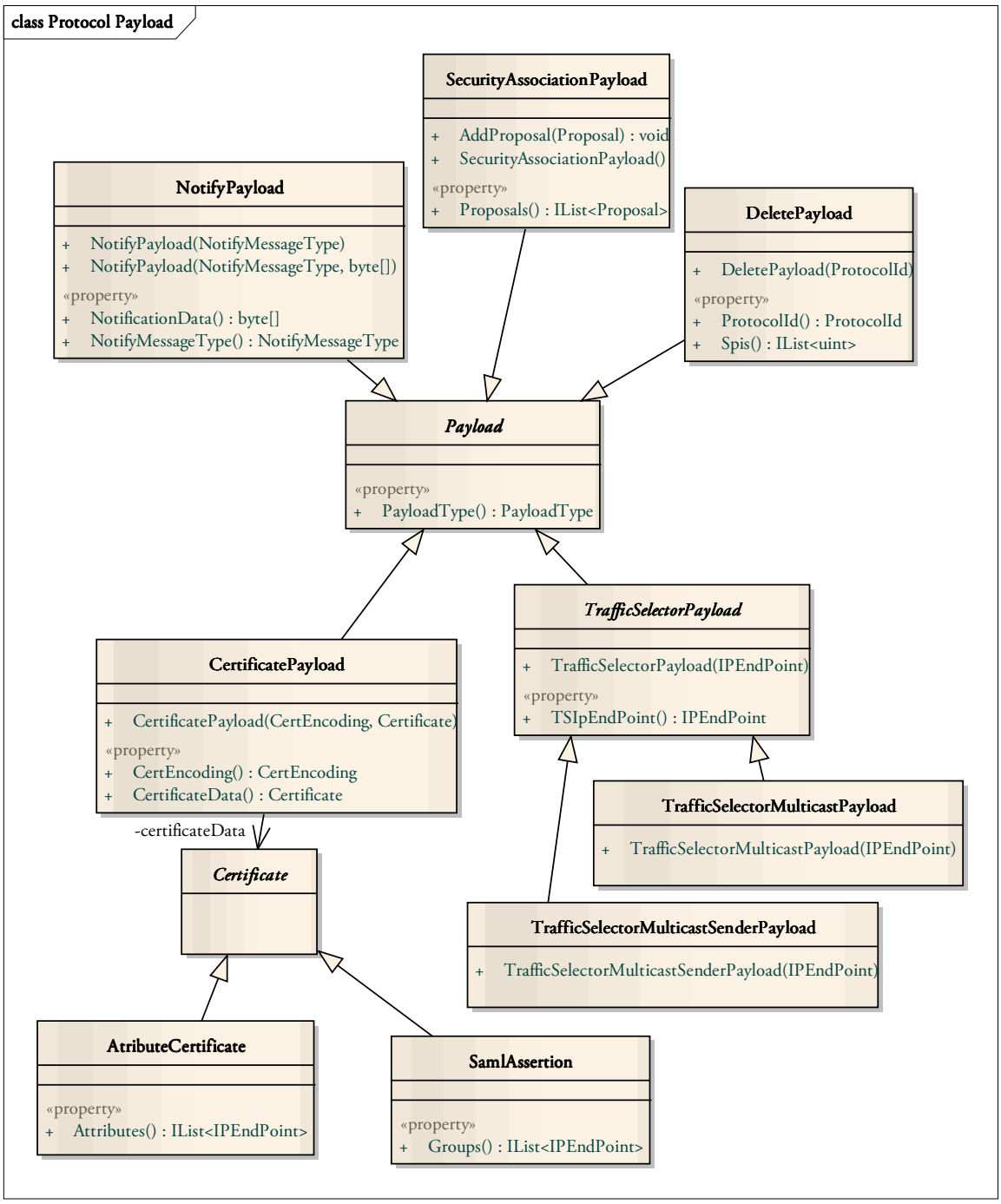

Abbildung 35: Klassen im Namespace IpSecMulticast.Protocol (2)

Die erstellten Payloads in diesem Namespace entsprechen den Payloads, welche im IKEv2 Standard [6] definiert wurden. In der Abbildung 35 sind die wichtigsten davon enthalten. Einige Payloads, welche für die Realisierung dieser IPsec Multicast Lösung keine entscheidende Rolle gespielt haben wurden weggelassen.

4.5 Designentscheide

Während der Entwicklung wurde versucht eine möglichst einfache Lösung zu erstellen, welche den Anforderungen entspricht. Die folgenden Abschnitte erklären einige spezielle Eigenheiten im Design der *IpSecMulticast*-Bibliothek.

4.5.1 Implizite GROUP_SA

Für das Bereitstellen eines GCKS muss eine Instanz der Klasse *IpSecMulticastService* mit folgendem Konstruktoraufruf erstellt werden:

```
IpSecMulticastService mcService =
    new IpSecMulticastService(IPAddress.Any, true);
```

Dabei gibt der erste Parameter an, dass auf allen lokalen IP-Adressen auf ankommende IKE-Nachrichten gewartet wird. Der zweite Parameter hat zur Folge, dass nicht ein normaler *SaManager* instanziiert wird, sondern eine Instanz des *GcksSaManager* dem Service hinzugefügt wird. Dadurch erhält der *IpSecMulticastService* die Funktionalität für das Verwalten der Subscriber und das Erstellen von GROUP_SAs.

Möchte man einen Subscriber, d.h. einen Sender oder Receiver erstellen, wird erneut die Klasse *IpSecMulticastService* verwendet. Der Konstruktoraufruf sieht dann folgendermassen aus:

```
IpSecMulticastService mcService =
    new IpSecMulticastService(IPAddress.Any, false);
```

Die Angabe von *IPAddress.Any* bewirkt nun, dass sich der Subscriber über die Default IP-Adresse mit dem GCKS verbindet. Wie bereits erwähnt bewirkt der zweite Parameter mit *false*, dass eine Instanz des *SaManagers* erstellt wird.

Mit dem nun zur Verfügung stehenden *IpSecMulticastService*-Objekt können bereits Multicast Daten empfangen oder veresendet werden. Dazu ist lediglich eine Angabe des IP-Endpunktes (IP-Adresse und Port) nötig. Der Service fordert dann implizit vom GCKS eine GROUP_SA an und sendet oder empfängt Daten über den angegebenen IP-Endpunkt mit der erhaltenen GROUP_SA.

```
byte[] message = ASCIIEncoding.ASCII.GetBytes("Hello world!");

IPEndPoint multicastIpEndPoint =
    new IPEndPoint(IPAddress.Parse("224.5.6.7"), 65000);

mcService.Send(message, multicastIpEndPoint);
```

Der erstellte *IpSecMulticastService* ist in der Lage sowohl als eigenständiger Subscriber beim GCKS über das Netzwerk neue GROUP_SAs anzufordern, wie auch als GKCS lokale GROUP_SAs anzulegen und über diese Daten zu transferieren. Im Folgenden wird der Ablauf der beiden Varianten erklärt.

Lokaler GCKS

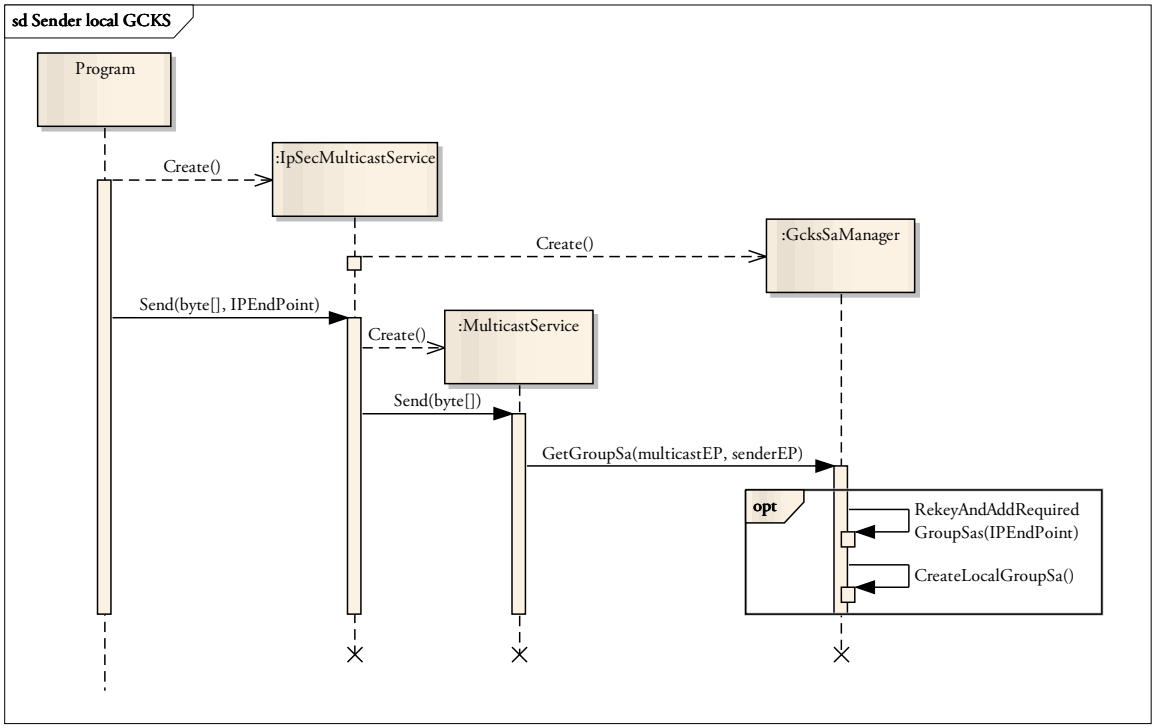

Abbildung 36: Sendevorgang mit lokalem GCKS

Nachdem die beiden Klassen *IpSecMulticastService* und *GcksSaManager* instanziiert wurden, gibt das Programm den Befehl zum Senden. Dies bewirkt, dass zum angegebenen Multicast IP-Endpunkt ein *MulticastService* erstellt wird. An diesen wird der Send-Befehl weitergeleitet.

Der *GcksSaManager* stellt die Methode *GetGroupSa()* zur Verfügung. Sie gibt eine GROUP_SA zu einem Multicast IP-Endpunkt, über den die Daten empfangen, und zu einem Sender IP-Endpunkt, über den die Daten gesendet werden, zurück.

Innerhalb der Methode wird überprüft, ob die Gruppe bereits lokal vorhanden ist. Ist dies nicht der Fall, werden die notwendigen GROUP_SAs zum angegebenen Multicast IP-Endpunkt rekeyed und der lokalen GROUP_SA-Liste hinzugefügt. Zusätzlich wird eine GROUP_SA erstellt, über die aktuell gesendet werden möchte.

Der *GcksSaManager* kann nun die geforderte GROUP_SA dem *MulticastService* zurückgeben, wo sie zum Verschlüsseln der Multicast-Daten verwendet wird. Anschliessend werden diese über das Netzwerk an die entsprechende Multicast-Destination übertragen.

Remote GCKS

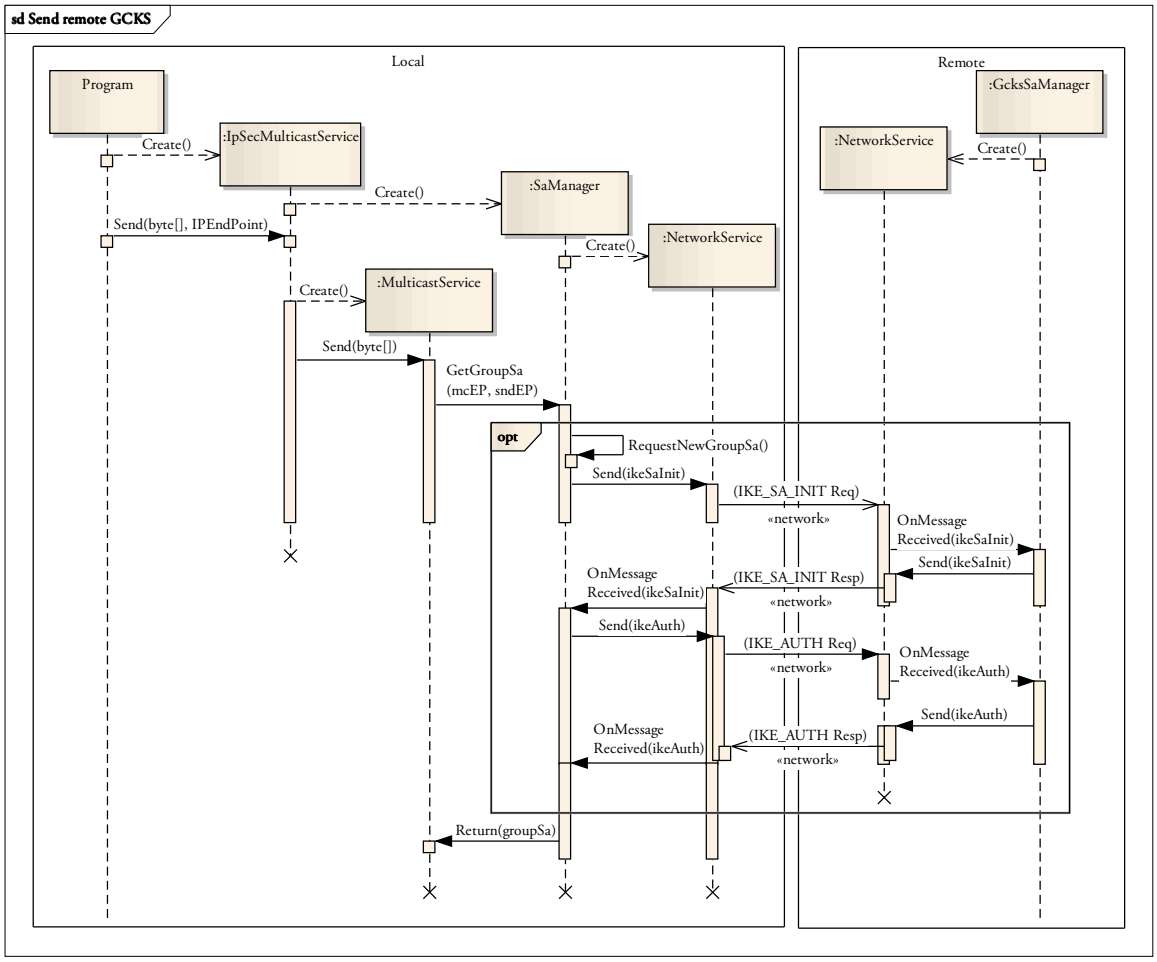

Abbildung 37: Sendevorgang mit remote GCKS

Die Variante unterscheidet sich bis zum Aufruf von *GetGroupSa()* lediglich dadurch, dass lokal anstatt eines *GcksSaManager* ein *SaManager* instanziiert wird. Wenn dieser die angeforderte GROUP_SA nicht enthält, muss sie mit *RequestNewGroupSa()* beim GCKS angefordert werden. Dazu werden über das Netzwerk die beiden Nachrichtenpaare IKE_SA_INIT und IKE_AUTH übertragen.

Der *SaManager* enthält nun die richtige GROUP_SA und kann diese dem *MulticastService* zurückgeben. Dieser verschlüsselt die Nutzlast-Daten mit der erhaltenen GROUP_SA und sendet sie über UDP an die entsprechende IP-Multicast Adresse.

4.5.2 State Pattern IkeSa

Nicht jeder IKE-Nachrichtentyp ist zu jedem Zeitpunkt erlaubt. Beispielsweise darf ein CREATE_CHILD_SA Request nicht verarbeitet werden, bevor die IKE_SA überhaupt erfolgreich erstellt wurde und mit IKE_AUTH eine Authentisierung stattgefunden hat.

Zur Lösung dieser Problemstellung wird auf einem *IkeSa* Objekt einen Status geführt. Dieser Status betrifft vor allem den Verbindungsaufbau. Während dieser Zeitspanne wird bei jedem Eintreffen einer Nachricht der Status geändert. Die Abbildung 38 zeigt das entsprechende Zustandsdiagramm.

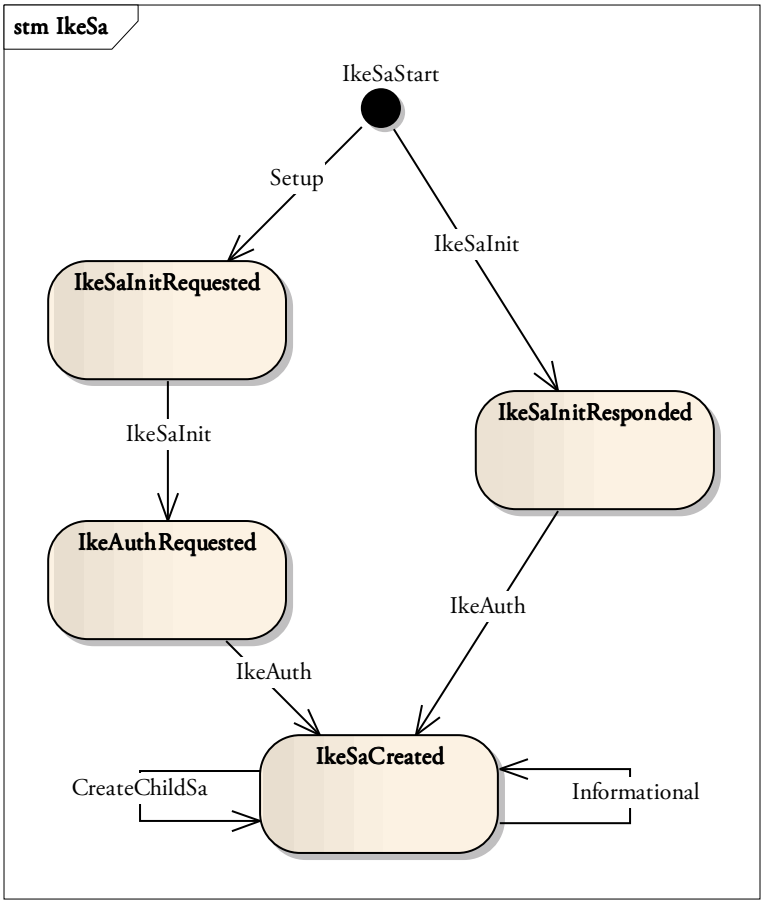

Abbildung 38: Zustandsdiagramm eines IkeSa-Objekts

Je nachdem welchen Status eine *IkeSa* hat, sind andere Nachrichtentypen erlaubt. Zur Überprüfung ob nun eine empfangene Nachricht gültig ist, implementiert die Klasse *IkeSa* die Methode *ValidateMessage()*. In dieser Methode wird die Überprüfung an den jeweils aktuellen Status delegiert.

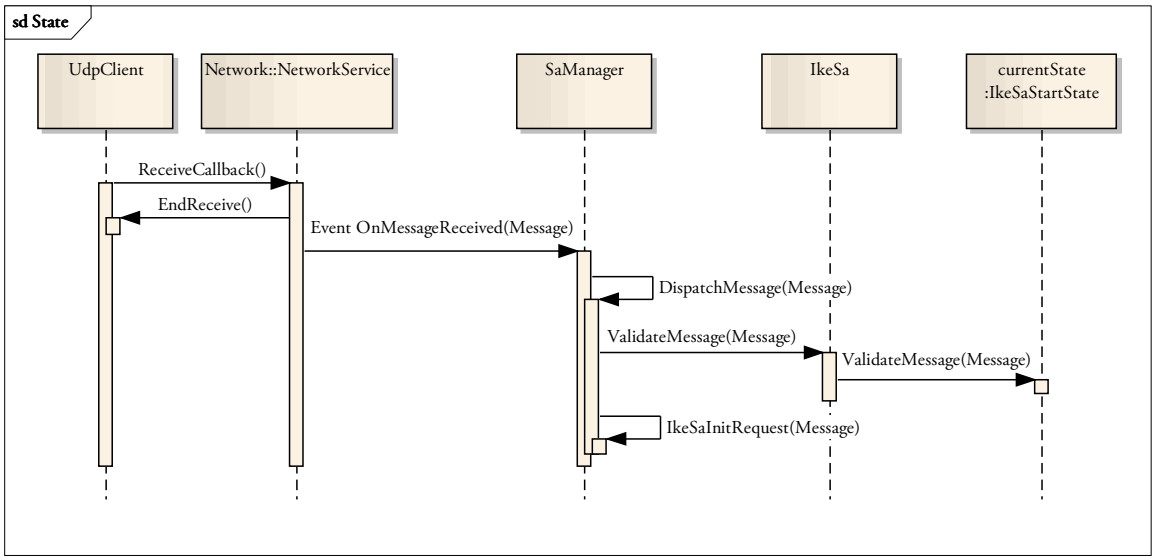

Abbildung 39: Sequenzdiagramm zur Validierung einer IKE-Nachricht

Das Sequenzdiagramm in der Abbildung 39 zeigt den Ablauf beim Empfangen einer Nachricht im Detail. Im Abschnitt *4.4.3 Namespace IpSecMulticast.SecurityAssociation* sind die verwendeten State-Klassen in der Abbildung 33 aufgeführt.

4.5.3 GCKS

Der GCKS kann in zwei unterschiedlichen Szenarien zum Einsatz kommen. Zum einen kann er als eigenständiger Server lediglich für die Verwaltung der GROUP_SAs zuständig sein. Dies bedeutet, dass sich alle Subscriber beim GCKS melden und eine GROUP_SA anfordern.

In einem zweiten Szenario kann der GCKS zugleich auch selber als Sender oder Receiver fungieren. Dafür kann er für sich lokal eine GROUP_SA anlegen und diese zum Senden oder Empfangen verwenden.

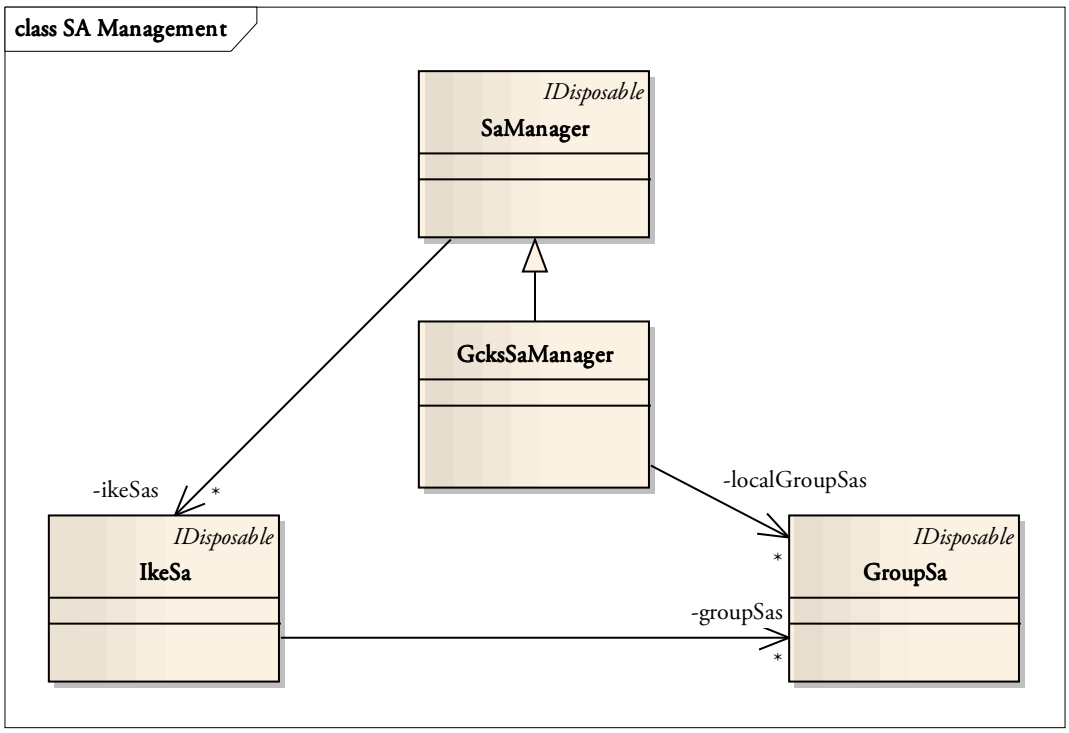

Abbildung 40: Klassen zur Verwaltung der IKE_SAs und GROUP_SAs

Der *SaManager* verwaltet sowohl für den GCKS wie auch für einen „normalen“ Subscriber die vorhandenen IKE_SAs. Die vorhandenen GROUP_SAs sind jeweils der entsprechenden IKE_SA zugeordnet. Dank dieser Struktur ist jederzeit ersichtlich, welcher Subscriber (IKE_SA) welche GROUP_SAs abonniert hat.

Da wie vorherig beschrieben der GCKS auch selber zum Sender oder Empfänger werden kann, verwaltet er seine lokal verwendeten GROUP_SAs in der Collection *localGroupSas*.

Die Logik für das Verarbeiten von Nachrichten ist im *SaManager* implementiert. Dieser überprüft den empfangenen Nachrichtentyp und leitet die Message an die entsprechende Methode weiter, welche diese Nachricht verarbeiten kann. Nun gibt es gewisse Nachrichten, welche nur ein Receiver oder Sender (= Subscriber) verarbeiten muss und gewisse Nachrichten, welche nur der GCKS verarbeiten muss.

Alle empfangbaren Nachrichten sind als Methode im *SaManager* vorhanden. Jene Nachrichten, die ein gewöhnlicher Subscriber verarbeiten können muss, sind auch dort implementiert. Eine Nachricht, welche nur der GCKS verarbeiten können muss, ist im *SaManager* zwar implementiert, löst aber einen Fehler aus, falls die Methode tatsächlich aufgerufen wird. Im *GcksSaManager* wird die entsprechende Methode, beispielsweise *IkeSaInitRequest()*, überschrieben und somit auch aufgerufen, falls ein IKE_SA_INIT Request empfangen wird.

Diese Lösung erlaubt die Verwendung des gleichen Codes für die Nachrichtenüberprüfung und Weiterleitung im GCKS wie auch in einem normalen Subscriber. Trotzdem wird aber sichergestellt, dass ein Subscriber nicht plötzlich fähig ist, Nachrichten zu verarbeiten, die eigentlich nur ein GCKS verarbeiten können sollte.

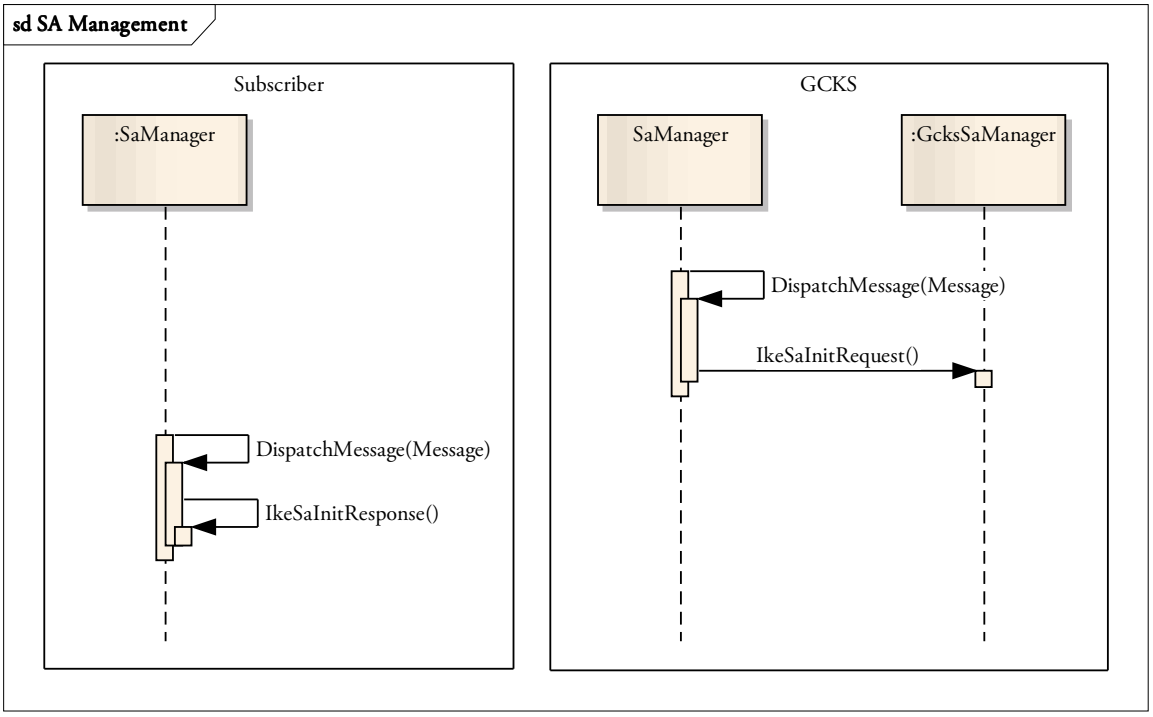

Abbildung 41: Verarbeitung eintreffender Nachrichten

Die Abbildung 41 zeigt nochmals die Verarbeitung einer eingehenden Nachricht. Auf der linken Seite wurde ein Objekt vom Typ *SaManager* instanziiert. Die Verarbeitung folgt direkt auf dem erstellten Objekt. Bei einem GCKS heisst die verantwortliche Klasse *GcksSaManager* und ist vom normalen *SaManager* abgeleitet. Wird eine entsprechende Nachrichte erhalten, wird die überschriebene Methode aus dem *GcksSaManager* aufgerufen.

4.5.4 Struktur Security Associations

Im IKEv2 Standard besteht zwischen zwei IPsec Endpunkten ein IKE_SA. Diese Security Association wird verwendet, um die Konfigurations- und Verwaltungsnachrichten zu sichern. Der Datenverkehr wird über sogenannte CHILD_SAs gesendet. Falls nun also Datenverkehr per ESP abgesichert werden soll, wird eine CHILD_SA für ESP zwischen den Peers ausgehandelt und im System-Kernel installiert. Wie der Name schon sagt wird die CHILD_SA als Kind an die entsprechende IKE_SA angehängt.

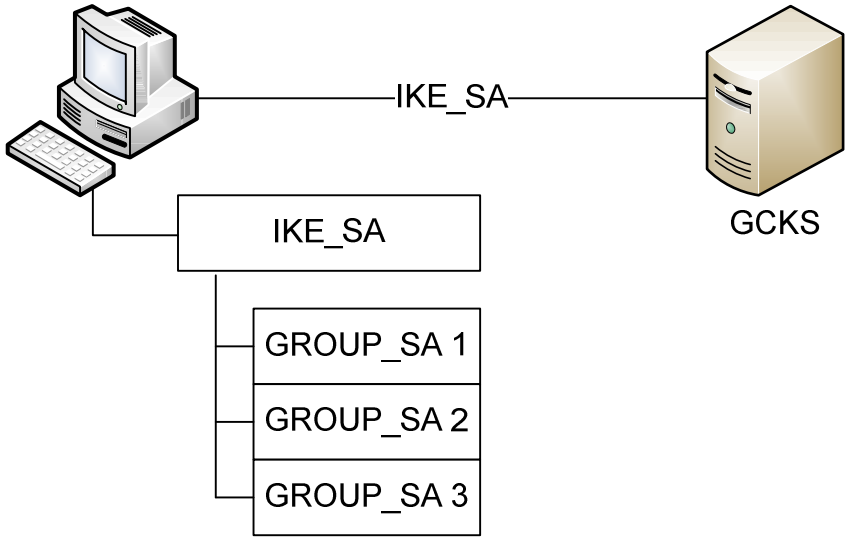

Abbildung 42: Struktur IKE_SA mit mehreren GROUP_SAs

Die entwickelte IPsec Multicast Lösung verwendet eine ähnliche Struktur der Security Associations. Wenn Multicast-Daten geschützt werden sollen, wird vom GCKS eine GROUP_SA angefordert. Diese wird ebenfalls an die entsprechende IKE_SA angehängt. In der Abbildung 42 ist dies konzeptionell ersichtlich.

Vor allem bei der Funktionalität des GCKS hat dies eine entscheidende Rolle. Denn dort werden sämtliche GROUP_SAs gemanaged. Sobald ein neuer Subscriber eine GROUP_SA anfordert, muss überprüft werden, ob allenfalls bereits eine entsprechende Security Association besteht. Dazu wird durch alle vorhandenen IKE_SAs durch iteriert und nach einer passenden GROUP_SA gesucht. Ist die Suche erfolglos, muss eine neue GROUP_SA angelegt und der anfragenden IKE_SA zugewiesen werden.

4.5.5 IKE-Messaging

Für das Versenden von IKE-Nachrichten wurden im Namespace *IpSecMulticast.Protocol* eine *Message* Klasse und die dazugehörigen *Payloads* implementiert. Diese Klassen sind nach dem Vorbild des IKEv2 Standards aufgebaut.

Wenn eine neue Nachricht gesendet werden muss, wird zuerst ein Objekt der Klasse *Message* erstellt. Anschliessend werden die notwendigen *Payloads* instanziiert und der *Message* Objekt hinzugefügt. Die Klasse *IkeMessageCreator* ist verantwortlich für das Erstellen einer Message und das Hinzufügen der benötigten Payloads.

Übertragen werden die Nachrichten in Form von serialisierten Objekten. Dies lässt sich sehr einfach realisieren. Zudem soll die erstellte Software hauptsächlich das erarbeitete Konzept überprüfen und somit ist diese einfache Art der Nachrichtenübertragung die beste Wahl.

4.5.6 Retransmission

Die Übertragung der IKE-Nachrichten wird über UDP durchgeführt. Dies hat zur Folge, dass die Datenübertragung nicht zuverlässig ist. Es können also Pakete verloren gehen und es kann zum Überholen von Paketen kommen.

Der IKEv2 Standard beschreibt, wie das Protokoll zuverlässig implementiert werden kann. Die Nachrichten sind jeweils in Request/Response Paaren vorhanden. Eine Response bestätigt jeweils einen Request. Der Sender eines Requests muss bei einem Timeout eine Retransmission durchführen. Ebenfalls ist definiert, wie die Window Size zu interpretieren ist.

Die erstellte Software implementiert diese zuverlässige Art der Nachrichtenübertragung ebenfalls. Die Logik dafür ist komplett in der Klasse *NetworkService* vorhanden. Da sich ein Request oder eine Response jeweils auf eine IKE_SA bezieht, sind die notwendigen Datenstrukturen für eine Retransmission in der Klasse *IkeSa* vorhanden. Die Abbildung 43 zeigt die relevanten Klassen, Methoden und Felder.

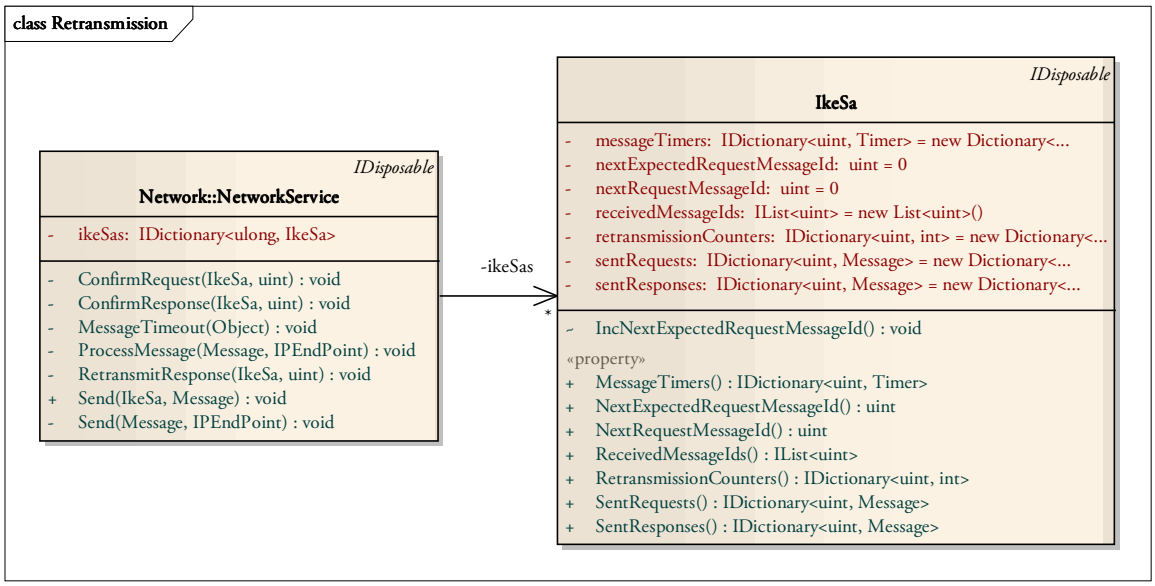

Abbildung 43: Datenstrukturen zur Durchführung einer Retransmission

Die Klasse *NetworkService* hat eine Referenz auf das gleiche Dictionary mit den IkeSa-Objekten darin wie der *SaManager*. Die gezeigten Felder und öffentlichen Properties der Klasse IkeSa werden in der Tabelle 7 kurz erklärt.

Property	Bedeutung
sentRequests	Alle ausgehenden Requests werden in dieser Struktur abgelegt, damit die Nachricht bei einem Retransmission-Vorgang erneut gesendet werden kann. Wird eine Response empfangen, kann der entsprechende Request gelöscht werden.
sentResponses	Die ausgehenden Responses werden ebenfalls für eine allfällige Retransmission temporär abgelegt.
nextRequestMessageId	Gibt die ID an, welche der nächste ausgehende Request haben wird.
nextExpectedRequestMessageId	Gibt die erwartete Message-ID an, die der nächste eintreffende Request haben sollte. Ist die ID eines eintreffenden Requests grösser als dieser Wert plus die Window Size, wird die Nachricht verworfen.
messageTimers	Für jeden ausgehenden Request wird ein Timer initialisiert, welcher eine Retransmission auslöst, falls der Request nicht mit einer Response bestätigt wurde.
receivedMessageIds	Da sich die empfangenen Message-IDs im Bereich der Window Size bewegen können, werden diese IDs in einer temporären Liste gespeichert, damit der Wert *nextExpectedRequestMessageId* korrekt gesetzt werden kann.
retransmissionCounters	Wird ein Message-Timeout ausgelöst, führt der *NetworkService* eine Retransmission durch. Die Anzahl der Sendeversuche wird in diesem Feld gespeichert. Übersteigt diese Anzahl den in der Konfiguration definierten Wert, wird die IKE_SA abgebaut.

Tabelle 7: Felder für Retransmission

4.5.7 Message Receiving

Das Empfangen von IKE-Nachrichten wurde asynchron gelöst. Schliesslich kann zu jedem beliebigen Zeitpunkt eine Nachricht eintreffen, was das Einsetzen eines eigenen Threads für das Empfangen von Nachrichten notwendig macht.

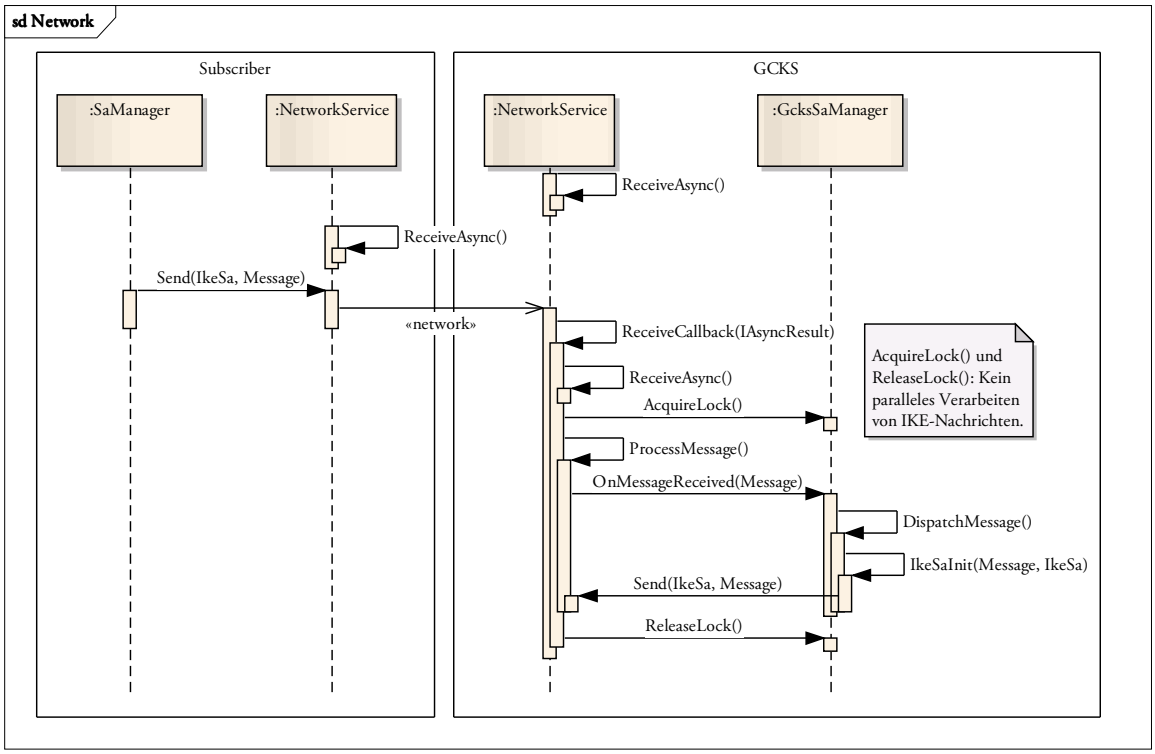

Abbildung 44: Sequenzdiagramm zum Empfangen einer Nachricht

Wenn ein Objekt der Klasse *NetworkService* erzeugt wird, ruft der Konstruktor die Methode *ReceiveAsync()* auf, welche einen Thread aus dem .NET Threadpool anweist, auf eine eintreffende Nachricht zu warten. Sobald eine solche ankommt, wird die Callback-Methode *ReceiveCallback()* ausgelöst. Damit weitere Nachrichten empfangen werden können, wird erneut mit *ReceiveAsync()* auf eine Nachricht gewartet.

Da der erstellte *SaManager* nicht auf die parallele Verarbeitung von Nachrichten ausgelegt wurde, wird dieser mit den Events *AcquireLock()* und *ReleaseLock()* für die exklusive Verwendung gesperrt. Nach dem Lock wird die IKE-Nachricht deserialisiert und im Erfolgsfall mit *ProcessMessage()* und dem Event *OnMessageReceived()* an den *SaManager* weitergeleitet.

Mit *DispatchMessage()* und der entsprechenden Nachrichten-Methode, in Abbildung 44 *IkeSaInit()*, verarbeitet der *SaManager* die Nachricht und sendet die erstellte Response über den *NetworkService* zurück an den Absender des Requests.

4.6 Technologien

4.6.1 log4net

Zur Überprüfung der Funktionalität, Suche von Fehlern und Ausgabe von Statusmeldungen wurde die Logging-Komponente log4net von Apache eingesetzt. Sie ist die .NET Äquivalente zur bekannten JAVA-Lösung log4j. Die umfangreiche Dokumentation ist auf der Website [10] von Apache zu finden.

Die folgende Tabelle 8 gibt an, in welchem Loglevel welche Arten von Nachrichten ausgegeben werden.

Log-Level	Ausgabe
Debug	Beim Senden oder Empfangen einer IKE-Nachricht, wird ein Logeintrag geschrieben. In diesem sind die SPIs, der Typ der Nachricht und allfällige Notifications enthalten.
Info	Jedes Mal, wenn eine Security Association erstellt, rekeyed oder gelöscht wird, schreibt die verantwortliche Komponente einen Logeintrag.
Error	Wenn während dem Programmablauf ein Fehler eintritt, welcher das Fortsetzen des Ablaufs unmöglich macht, wird dies auf diesem Loglevel eingetragen.

Tabelle 8: Erklärung der Loglevels

4.6.2 LINQ

Der .NET Namespace System.Linq enthält Funktionalität für die einfachere Verarbeitung von Collections. Überall, wo eine Vereinfachung des Codes mit Hilfe von LINQ Funktionalität erzielt werden konnte, wurde diese Linq-Syntax eingesetzt. Der folgende Code-Auszug zeigt ein Beispiel dieser Syntax.

```
var existingGroupSas = from groupSa in ikeSa.GroupSas.Values
                       wherea groupSa.MulticastIpEndPoint.Equals(
                           multicastIpEndPoint)
                       select groupSa.Spi;
```

Dabei werden aus einer Liste von GROUP_SAs diejenigen selektiert, bei welchen das Property *MulticastIpEndPoint* mit dem übergebenen *multicastIpEndPoint* übereinstimmt. Das Resultat ist eine vollständig typensichere Collection, welche die entsprechenden Referenzen enthält.

4.6.3 Events

Für eine Reduktion der Kopplung zwischen zwei Softwareklassen können Events eingesetzt werden. Diese entsprechen der Observer-Technik. Ein Objekt kann sich bei einem anderen Objekt registrieren und wird benachrichtigt, wenn der Event ausgelöst wird.

Solche Events wurden vor allem in der Klasse *NetworkService* eingesetzt, damit nur eine einseitige Kopplung zum *SaManager* besteht. Das folgende Beispiel soll die Funktionsweise verdeutlichen.

Code im NetworkService

```
// Definition
internal delegate void MessageReceivedHandler(Message m, IPEndPoint
    remoteIpEndPoint);
public event MessageReceivedHandler OnMessageReceived;

// Event auslösen
if (OnMessageReceived != null) OnMessageReceived(m, remoteIpEndPoint);
```

Code im SaManager

```
// Registrierung beim Event
this.networkService.OnMessageReceived += this.OnMessageReceived;

// Event-Methode
private void OnMessageReceived(Message m, IPEndPoint remoteIpEndPoint)
{
    // Code für die Verarbeitung
}
```

4.7 Codedokumentation

Der Sourcecode dieses Projekts wurde komplett mit XML-Kommentaren dokumentiert. Der Export dieser Code-Dokumentation ist im Projektordner beziehungsweise auf der Projekt-CD unter *21 Codedokumentation* zu finden.

5 Simulator

5.1 Portzuweisung

Für die Kommunikation zweier IKE-Partner wird beim IKEv2 standardmässig die Port-Nummer 500 oder bei Verwendung von NAT die Port-Nummer 4‘500 verwendet. Um das Debugging des Simulators zu erleichtern, soll er mehrfach auf einer Maschine gestartet werden können. Damit ein GCKS und mehrere Subscriber gleichzeitig gestartet werden kann, muss die Port-Zuweisung dynamisch erfolgen. Der IKE- und Multicast-Verkehr wird dabei unterschiedlich behandelt.

IKE-Verkehr

Für die Unicast-Kommunikation über die IKE_SA wird dem GCKS stets den Port 60‘500 zugewiesen. Die Port-Zuweisung beim Subscriber erfolgt durch das System. Empfängt der GCKS eine Nachricht auf dem Port, kann er anhand der Absender-Informationen den Port des Subscribers auslesen.

Multicast-Verkehr

Der Multicast-Verkehr wird vom Subscriber auf einem bestimmten Port (bspw. 65‘000) empfangen. Die Port-Zuweisung auf dem Sender erfolgt dynamisch. Dies hat zur Folge, dass auf einer Maschine mehrere Sender für einen Port jedoch nur ein Empfänger gestartet werden kann.

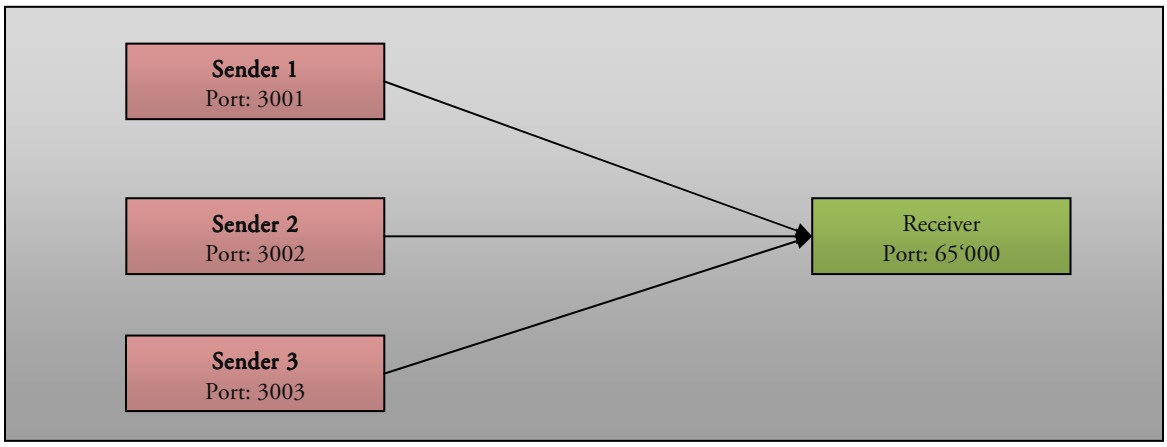

Abbildung 45: Portzuweisung Multicast-Verkehr

5.2 Funktionsumfang des Simulators

Während der Konzeptphase dieses Projekts stand vor allem ein Szenario einer IPsec Multicast Umgebung im Vordergrund. Dabei war von einem GCKS, der zugleich auch Sender der Multicast-Gruppe sein sollte, und mehreren Receivern die Rede. Die Abbildung 46 verdeutlicht dieses Szenario.

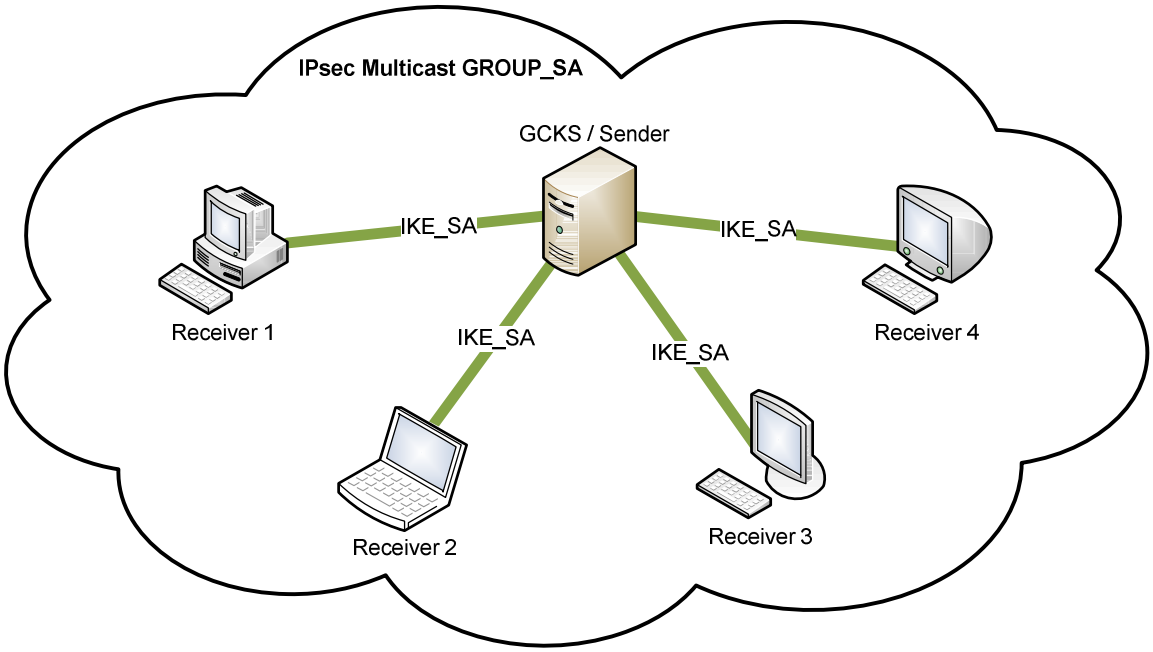

Abbildung 46: Szenario mit einem Sender als GCKS

Zu einem späteren Zeitpunkt im Projekt versuchte man die Komponenten Sender, Receiver und GCKS voneinander zu trennen, um auch das zweite Szenario realisieren zu können. In diesem sind GCKS, Sender und Receiver völlig voneinander entkoppelt. Dadurch wird ein IPsec Multicast Teilnehmer viel flexibler und kann zugleich Sender und Receiver sein.

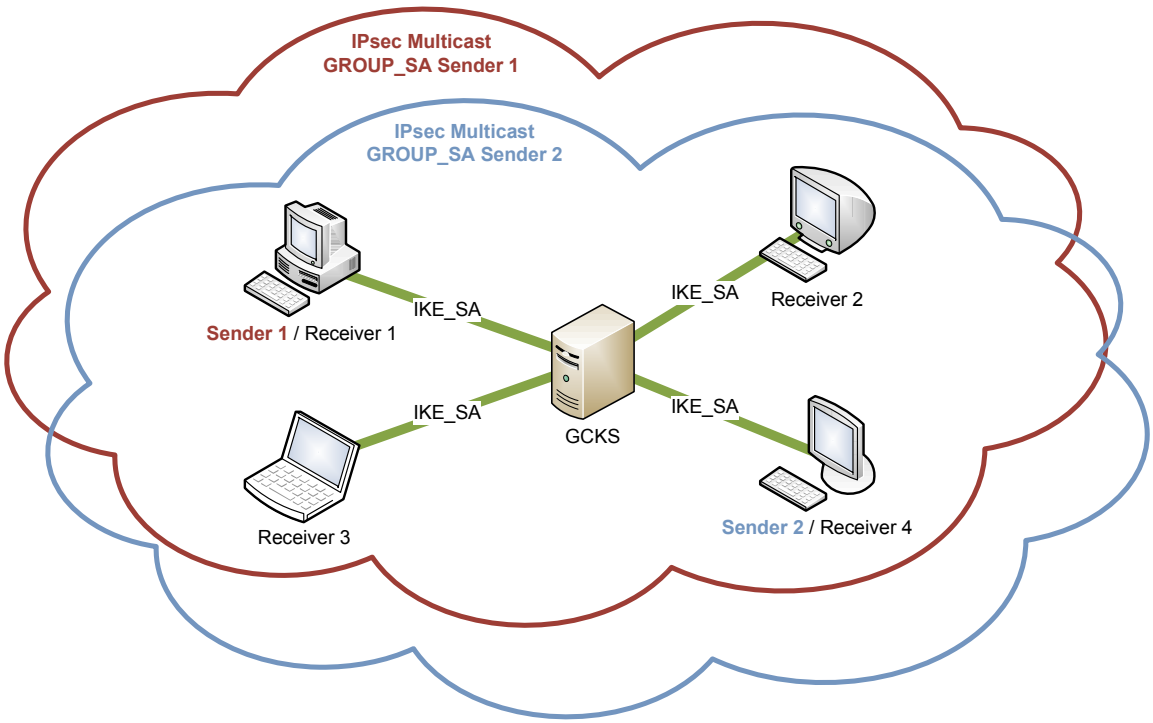

Abbildung 47: Szenario mit entkoppeltem GCKS

Die Abbildung 47 zeigt eine IPsec Multicast Umgebung mit einem GCKS und zwei Sendern. Pro Sender wurde eine GROUP_SA angelegt, die an alle Teilnehmer verteilt wurde. Somit kann ein Receiver zugleich zum Sender werden.

5.2.1 Lokaler GCKS

Die Schwierigkeit bei der Realisierung des Szenario 1 (siehe Abbildung 46) war die Funktionalität für einen lokalen GCKS. Ein Sender musste also zugleich auch GCKS sein können. Nachdem die einzelnen Komponenten Receiver, Sender und GCKS entkoppelt waren, mussten sie also wieder möglichst einfach zusammengefügt werden.

Ebenfalls musste dem GCKS weitere Logik hinzugefügt werden, welche es ihm erlaubt, seine lokal verwendeten Gruppen (als Sender oder Receiver) zu verwalten. Dabei müssen die GROUP_SAs entsprechend angepasst werden, wenn ein neuer Sender zur Gruppe hinzustösst, ein Rekeying stattfindet oder ein Sender die Gruppe verlässt.

5.2.2 Multi-Sender

Ein Szenario mit mehreren Sendern, wie es in Abbildung 47 zu sehen ist, hat einen weitaus grösseren Verwaltungsaufwand, als wenn nur ein einzelner Sender pro Multicast-Gruppe zugelassen ist. Wie im

Konzept (siehe *3.3.9 Multi-Sender-Architektur*) beschrieben wird pro Sender eine eigene GROUP_SA angelegt, welche an alle Teilnehmer der IPsec Multicast Gruppe verteilt wird.

Wenn ein neuer Teilnehmer sich beim GCKS anmeldet und eine GROUP_SA anfordert, muss erkennbar sein, ob der Subscriber der Gruppe als Sender oder nur Receiver beitreten möchte. In beiden Fällen werden sämtliche GROUP_SAs zu dieser Multicast-Gruppe rekeyed und dem neuen Subscriber mitgeteilt. Der Unterschied zwischen einem neuen Sender und einem Receiver liegt darin, dass für einen Sender eine zusätzliche Gruppe erstellt und diese an alle Teilnehmer propagiert werden muss.

5.2.3 Testumgebung

Die entwickelte Testumgebung, wie sie im Abschnitt *6 Tests* beschrieben ist, hat viel zu einer guten Softwarelösung beigetragen. Während der gesamten Projektphase sind stets wieder Spezialfälle aufgefallen, welche später als Testszenario realisiert und geprüft wurden. Dies hat bei der Entwicklung und vor allem bei sämtlichen Refactoring-Arbeiten eine grosse Sicherheit gegeben.

Da es sich bei diesem Projekt nicht um ein Softwareprojekt im klassischen Sinne handelt, war auch schon zu Beginn klar, dass „normale“ Unit-Tests keine grosse Anwendung finden werden. Deshalb hat die Entwicklung der Testumgebung, bei welcher mehrere Prozesse je nach Testszenario als GCKS, Receiver oder Sender gestartet werden können, einen grossen Mehrwert für die erstellte Software.

5.3 Limitierungen des Simulators

Im Wesentlichen war das Ziel dieses Projekts ein Konzept für IPsec Multicast zu erstellen, sowie dessen Realisierung in einem Simulator zu testen. Dabei wurde nie versucht eine komplette IKEv2 Lösung, wie beispielsweise strongSwan, nachzubauen. Entsprechend wurden viele Funktionen, welche in strongSwan schon implementiert sind, nicht in den erstellten Simulator eingebaut.

Des Weiteren existieren noch einige Limitierungen, welche eine praxistaugliche IPsec Multicast Lösung nicht haben dürfte. Diese Einschränkungen wurden im Rahmen dieses Projekts zum einen aus Zeitgründen nicht implementiert. Zum andern handelt es sich hierbei um einen Simulator, der die Stärken und Schwächen des Konzepts ersichtlich machen soll, und nicht um ein Endprodukt.

Die folgenden Abschnitte stellen eine Übersicht der wichtigsten Limitierungen des entwickelten Simulators dar.

5.3.1 Keine Adress- und Port-Ranges

In IKEv2 kann ein ganzer Range von IP-Adressen und Ports mit einer Security Association geschützt werden. Die Traffic Selectors für die Auswahl des Ranges sind dafür ausgelegt. Der erstellte Simulator ist nicht für Ranges ausgelegt. Er schützt jeweils eine Multicast IP-Adresse plus einen dazugehörigen Port mit einer eigenen GROUP_SA.

Diese Limitierung kommt daher, dass ein Subscriber beim Empfangen oder Senden auch definieren muss, auf welcher Multicast IP-Adresse und welchem Port er Daten empfangen oder senden möchte. Dieser IP-Endpunkt wurde dann weiterverwendet für die Traffic Selectors, welche zur Auswahl des Multicast-Traffic, der verschlüsselt oder entschlüsselt werde soll, verwendet werden.

5.3.2 High-Level Autorisierung

Die Autorisierung eines neuen Teilnehmers ist auf verschiedene Arten möglich. Gemäss den Anforderungen (siehe *3.2 Anforderungen*) soll sie über eine Policy-Datenbank, über Attributzertifikate, bzw. über SAML-Assertions möglich sein. Diese verschiedenen Möglichkeiten wurden zwar implementiert, aber nur auf einer primitiven Ebene.

Für die Überprüfung der Rechte über eine Policy-Datenbank wurde die lokale Konfiguration verwendet, in welcher die erlaubten Subscriber eingetragen werden können.

Eine Autorisierung über Zertifikate oder SAML-Assertions ist nur symbolisch möglich. Ein neuer Subscriber erstellt ein Objekt, welches einem Zertifikat oder einer SAML-Assertion entsprechen soll. Diesem Objekt fügt er die Multicast-Gruppen hinzu, für welche er eine GROUP_SA beantragen möchte. Das erstellte Zertifikats-Objekt wird mit dem IKE_AUTH Request übertragen. Der GCKS überprüft lediglich die enthaltenen Gruppen, um den neuen Teilnehmer zu autorisieren.

Die ausführliche Implementierung der Autorisierung hätte wenig Sinn gemacht, da strongSwan bereits viel Funktionalität für diese Verfahren bereitstellt. Beispielsweise das Erstellen und Parsen von Attributzertifikaten wird bereits unterstützt und eine Neuimplementierung für diesen IPsec Multicast Simulator wäre sehr umfangreich gewesen.

5.3.3 Autorisierung des Senders

Eine weitere Limitierung der Software ist, dass Sender und Receiver bei der Autorisierung nicht unterschieden werden. Die Trennung von Teilnehmern beim Gruppen-Beitritt würde durchaus Sinn machen. Dadurch könnte die Anzahl der Sender in einer Multicast-Umgebung unter Kontrolle behal-

ten werden. Je nach Konfiguration der Autorisierung wäre dann beispielsweise nur noch ein Sender pro IPsec Multicast Gruppe zugelassen.

Da aber die Autorisierung sowieso nur symbolisch implementiert wurde und nicht essentiell für die Überprüfung des erstellten Konzepts war, wurde die Unterscheidung von Sender und Receivern bei der Autorisierung weggelassen.

5.4 Anleitung Installation und Konfiguration

5.4.1 Bibliothek IpSecMulticast.dll

Die Bibliothek *IpSecMulticast* enthält die benötigte Funktionalität für die Erstellung einer IPsec Multicast Applikation. Die Details zur Implementierung sind im Abschnitt *4 Design* erläutert.

Konfiguration

Wird eine Instanz des *IpSecMulticastService* erstellt, greift der Service auf die Konfigurationsdaten zu. In der Konfigurationsdatei *IpSecMulticast.dll.config* können Grundwerte für den Betrieb des IPsec Multicast Teilnehmers festgelegt werden. Folgende Tabelle enthält Erklärungen zu den verschiedenen Einstellungen.

Einstellung	Beschreibung	Standard-Wert
GcksIp	IP-Adresse des GCKS	127.0.0.1
GcksPort	Port für die Unicast-Kommunikation zwischen dem GCKS und dem Teilnehmer	60500
IkeSaLifetime	Die Lebenszeit in Sekunden einer IKE_SA. Nach Ablauf der Lebenszeit wird ein Rekeying durchgeführt.	20
GroupSaLifetime	Die Lebenszeit in Sekunden einer GROUP_SA. Nach Ablauf dieser Zeit wird vom GCKS ein Rekeying initiiert.	9
SubscriberId	Die Identifikation des Teilnehmers.	hmuster
MessageTimeout	Dauert es länger als das definierte Message-Timeout, bis die Response auf ein Request erhalten wird, übermittelt der Teilnehmer den Request erneut. Der Wert ist in Sekunden angegeben.	3

MessageRetransmissions	Anzahl der Versuche für eine erneute Übermittlung der Nachrichten.	2
WindowSize	Die Fenstergrösse für die Nachrichten Übermittlung. Definiert die Anzahl Nachrichten die maximal versandt werden können, bevor eine Bestätigung erhalten wird.	10
SendCertificate	Angabe, welches Zertifikat verwendet werden soll. *SAML* für SAML-Assertion, *Attribute* für Attribut-Zertifikat oder *Local* für die lokalen Einstellungen kann als Wert eingegeben werden.	SAML
SubscriberSecret	Passwort des Teilnehmers zur Authentisierung.	password
PermittedGroups	Lokale Liste der bewilligten Gruppen (mehrere Gruppen durch Semikolon getrennt).	224.5.6.7:65000
DeadPeerDetectionTimeout	Anzahl Sekunden, die ohne eine Nachricht gewartet werden kann, bevor der Teilnehmer die Kommunikation zum GCKS überprüft.	15

Tabelle 9: Einstellungsmöglichkeiten der IpSecMulticast.dll

5.4.2 Applikationen

Durch Verwendung der *IpSecMulticast*-Bibliothek werden vier verschiedene Applikationen erstellt. Dessen Funktion wird im Folgenden kurz erläutert. Die Konfiguration wird aus der Bibliothek entnommen.

ProgramGcks.exe

Das Programm startet einen *IpSecMulticastService* in der Funktion eines GCKS. Dieser wartet auf Anfragen von Teilnehmern, welche Schlüsselinformationen zu den verschiedenen GROUP_SAs benötigen. Der GCKS ermittelt nicht nur die Schlüsselinformationen der GROUP_SAs sondern führt ebenfalls das Rekeying der GROUP_SAs durch.

ProgramSender.exe

Durch den Start der Applikation wird ein *IpSecMulticastService* als gewöhnlicher Teilnehmer gestartet. Nach dem Anmelden an den GCKS erhält er die Schlüsselinformationen zur Multicast-Gruppe 224.5.6.7:65000 und sendet im Abstand von 3 Sekunden verschlüsselte Multicast-Nachrichten ins Netzwerk.

ProgramReceiver.exe

Als Gegenstück zur Sender- Applikation fungiert die Receiver-Applikation. Ebenfalls wird der Service als gewöhnlicher Teilnehmer gestartet. Sobald die Schlüsselinformationen erhalten wurden, wird in einer Endlosschleife auf Nachrichten der Multicast-Gruppe 224.5.6.7:65000 gewartet, die Nutzdaten entschlüsselt und auf der Konsole ausgegeben.

Simulator.exe

Durch den Start des Simulators wird ebenfalls ein Service der *IpSecMulticast*-Bibliothek gestartet. Wird der Parameter „gcks“ mitgegeben, so wird der Service die Funktion eines GCKS übernehmen. Nach erfolgreichem Starten, wartet die Applikation auf die Eingabe von Kommandos. Damit kann der Anwender Nachrichten senden und empfangen und andere grundlegenden Funktionen aufrufen.

Die Beschreibung der Befehle wird in folgender Tabelle beschrieben.

Befehl	Beschreibung
s [3] 224.5.6.7:65000 Hallo Welt	Die Nachricht "Hallo Welt" wird dreimal mit dem Schlüsselmaterial der GROUP_SA 224.5.6.7:65000 verschlüsselt und danach an die Teilnehmer versandt. Der Gruppenbeitritt wird falls benötigt implizit ausgeführt.
r [3] 224.5.6.7:65000	Der Teilnehmer wartet dreimal nacheinander auf Multicast-Nachrichten mit der IP-Adresse 224.5.6.7 und dem Port 65000. Nach Erhalt werden die Nutzdaten entschlüsselt und auf der Konsole ausgegeben. Der Gruppenbeitritt wird falls benötigt implizit ausgeführt.
l 224.5.6.7:65000	Benötigt ein Teilnehmer das Schlüsselmaterial einer Gruppe nicht mehr, kann er sich damit von der GROUP_SA abmelden.
log 0 | 1 | 2	Dadurch kann der verwendete Log-Level der Applikation definiert werden. 0 Log ist deaktiviert 1 Log-Level auf INFO-Meldungen beschränkt 2 Log-Level DEBUG. Alle Log-Nachrichten werden ausgegeben.
show	Listet dem Teilnehmer alle verwendeten IKE_SAs mit den dazugehörigen GROUP_SAs auf.
quit	Löscht alle IKE_SAs und GROUP_SAs und beendet abschliessend die Applikation

Tabelle 10: Befehlsreferenz für Simulator.exe

Konfiguration log4net

Die Konfiguration des Log-Services log4net kann über die entsprechenden XML-Dateien der Applikationen verändert werden. Details zur Konfiguration sind auf der Homepage [10] des Herstellers aufgeführt.

6 Tests

6.1 Methodik

Für die Überprüfung der Funktionalitäten des Multicast-Simulators wurde eine Testumgebung entwickelt. Diese ermöglicht es in Form von Unit-Tests die einzelnen Szenarien zu durchlaufen. Innerhalb eines Tests werden verschiedene Prozesse des Simulators gestartet, Operationen ausgeführt und deren Resultate überprüft. Basierend auf den Rückgabewerten der Prozesse wird das Testresultat ermittelt. Nach fehlerfreiem Senden, Empfangen und Abmelden beim GCKS wird die Applikation beendet und der Erfolg des Szenarios gemeldet.

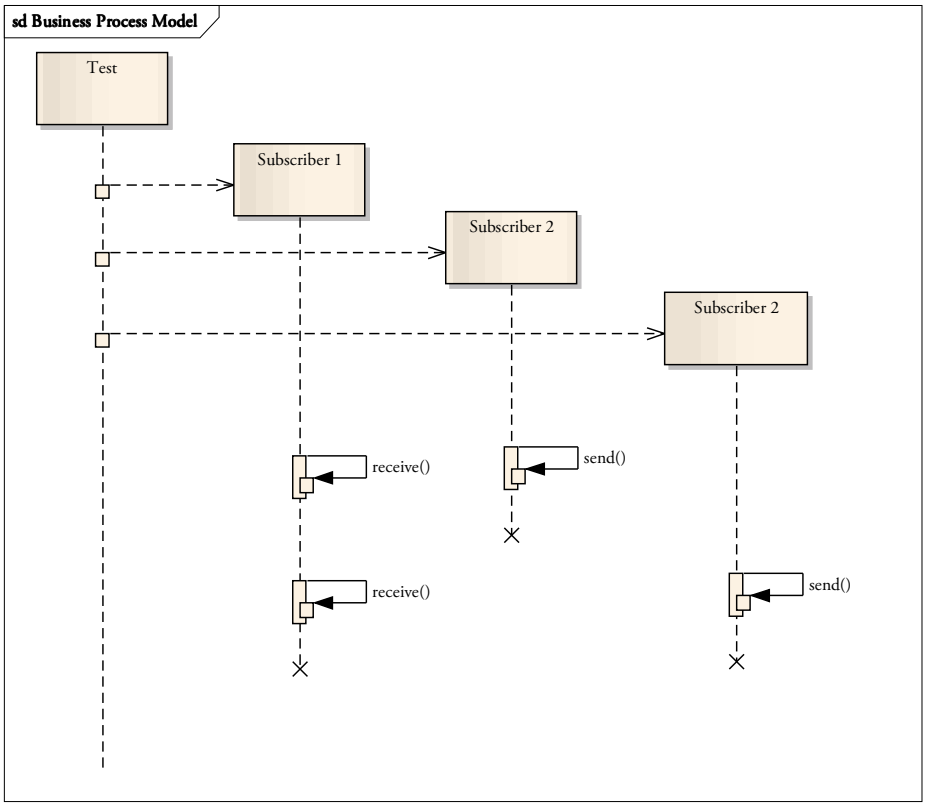

Abbildung 48: Testmethodik

Die Testumgebung wurde in C# entwickelt und lässt sich direkt aus dem Visual Studio analog zu Unit-Tests starten. Das Empfangen einer Multicast-Nachricht bewirkt, dass der Socket auf die IP-Adresse und einen Port gebunden wird. Da dies auf einem Windows-Betriebssystem pro Adress-Port-Kombination nur einmal möglich ist, mussten einige Tests auf zwei PCs aufgeteilt werden.

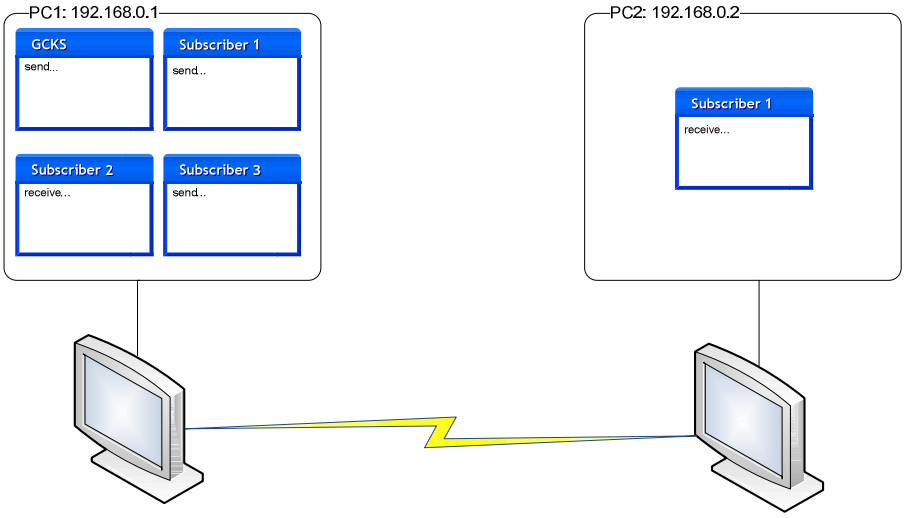

Abbildung 49: Aufbau der Testumgebung

6.2 Testfälle

Insgesamt wurden mit der oben beschriebenen Testmethodik 30 verschiedene Szenarien durchgetestet. Befanden sich mehrere Receiver in einem Szenario musste der Test wie erwähnt auf verschiedene Rechner aufgeteilt werden. Im Folgenden werden vier grundlegende Testszenarien kurz erläutert. Für eine detailierte Auflistung der Testszenarien sei auf das Dokument Testszenarien.xlsx verwiesen.

6.2.1 Test 1: Beitritt einer Multicast- Gruppe

Die Basisfunktion des Multi-Sender-Simulators wurde in einem einfachen Test überprüft. Zu den Basisfunktionen gehören der Beitritt zu einer Gruppe und das Senden bzw. Empfangen von verschlüsselten Nachrichten über IP Multicast. Nach dem Start der drei Komponenten wartete der Subscriber 1 blockierend auf eine erste Nachricht. Sobald vom Subscriber 2 eine Nachricht verschlüsselt versandt wurde, konnte diese vom Subscriber 1 empfangen und anschliessend entschlüsselt werden.

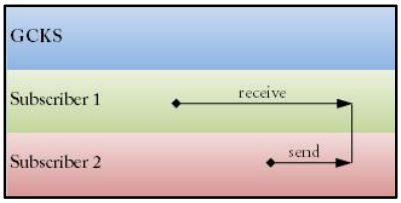

Abbildung 50: Test 1

Erfolgsfall

Entsprachen die Nutzdaten der empfangenen Nachricht den versandten Nutzdaten, wurde der Test erfolgreich abgeschlossen.

6.2.2 Test 5: Rekeying der IKE_SA und GROUP_SA

Damit das Rekeying der IKE_SA und der GROUP_SA im Betrieb getestet werden konnte, wurden ein sendender und ein empfangender Teilnehmer benötigt. Während der Subscriber 1 zwanzig Mal die Receive-Methode ausführte, sendete Subscriber 2 zwanzig Nachrichten per Multicast durch das Netzwerk. Während die Nachrichten zwischen den zwei Teilnehmern ausgetauscht wurden, führte der GCKS ein Rekeying der GROUP_SA und der IKE_SA durch. Damit wurde die lückenfreie Nachrichtenübermittlung auf die Probe gestellt.

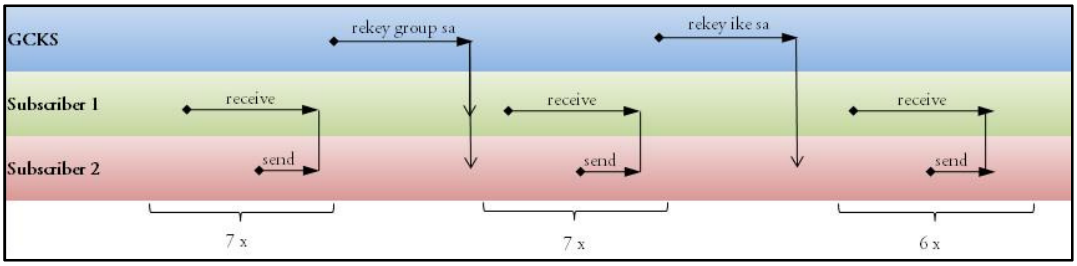

Abbildung 51: Test Rekeying

Erfolgsfall

Gelang es dem Subscriber 1 alle zwanzig Nachrichten zu empfangen und zu entschlüsseln, war der Test erfolgreich.

6.2.3 Test 7: Bei- und Austritte zu einer Multicast-Gruppe

Ein Teilnehmer kann sich nicht nur bei einer Gruppe anmelden, sondern er hat auch die Möglichkeit sich wieder abzumelden. Ob das Abmelden und Anmelden bei einer Gruppe problemlos funktioniert, wurde im siebten Test überprüft. Nach dem Empfang der ersten Nachricht meldete sich der Subscriber 1 aus der Multicast-Gruppe ab. Mittels Aufruf der Receive-Methode meldete sich der Subscriber 1 erneut beim GCKS an und erhielt die Schlüsselinformationen. Dies ermöglichte ihm die Daten von Subscriber 2 nach dem Erhalt zu entschlüsseln.

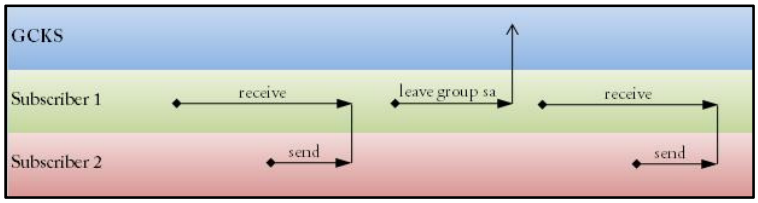

Abbildung 52: Bei- und Austritte zu einer Multicast-Gruppe

Erfolgsfall

Gelang es dem Subscriber 1 nach dem erneuten Eintritt in die Multicast-Gruppe die Daten des Subscriber 2 zu entschlüsseln, wurde der Test erfolgreich absolviert.

6.2.4 Test 14: Mehrere Sender innerhalb einer Multicast-Gruppe

Befinden sich mehrere Sender innerhalb einer Multicast-Gruppe, muss für jeden eine eigene GROUP_SA erstellt werden. Im vierzehnten Test wurde überprüft, ob der Subscriber 3 fähig ist, das korrekte Schlüsselmaterial anhand der Identifikation des Multicast-Verkehrs auszuwählen. Dazu senden nacheinander der GCKS, Subscriber 1 und Subscriber 2 ein verschlüsseltes Datenpaket an die Multicast-Gruppe.

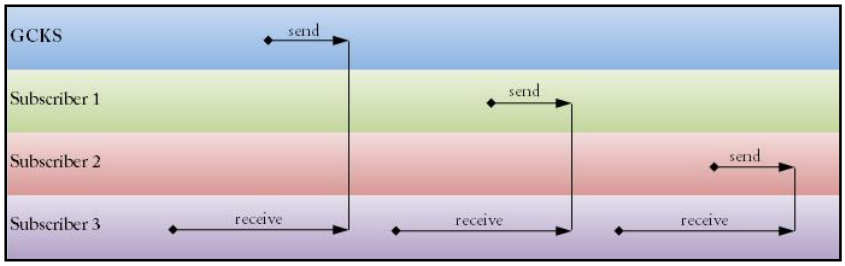

Abbildung 53: Mehrere Sender innerhalb einer Multicast-Gruppe

Erfolgsfall

Gelang es dem Subscriber 3 jeweils das richtige Schlüsselmaterial auszuwählen und damit die Daten zu entschlüsseln, wurde der Test erfolgreich absolviert.

7 Projektaussicht

Das Ziel dieses Projekts war die Entwicklung des Konzepts sowie des dazugehörigen Proof-of-Concept Simulators. Die Anforderungen der Aufgabenstellung konnten erfüllt werden. Dies war der erste Schritt in Richtung eines standardisierten IPsec Multicast Protokolls. Mögliche weiterführende Projekte werden in den folgenden Abschnitten erläutert.

Integration in strongSwan

Ein Fernziel des Betreuerteams ist die Integration einer IPsec Multicast Lösung in die bestehende IPsec-Lösung strongSwan. Das Resultat dieser Arbeit soll als Grundlage für die Implementierung verwendet werden. Da die Software-Architektur des Simulators nicht an den Aufbau von strongSwan angelehnt wurde, sind sicherlich einige Anpassungen in diesem Bereich nötig.

Autorisierung

Die Autorisierung der Teilnehmer wurde nur auf symbolischer Ebene implementiert. Die beiden Autorisierungs-Mechanismen SAML-Assertions und Attributzertifikate müssen vor einer Verwendung noch genauer untersucht werden.

TESLA

Die Authentisierung des Senders in einer IPsec Multicast Umgebung ist nach wie vor ein Problem. Da alle Teilnehmer einer Gruppe denselben Schlüssel verwenden, können Pakete unbemerkt gefälscht werden. Das TESLA-Protokoll könnte Abhilfe schaffen und in das erarbeitete Konzept integriert werden.

Erstellung Internet Draft

Um das Konzept publik zu machen und Expertenmeinungen einzuholen, kann die Erstellung eines Internet Drafts in Betracht gezogen werden.

8 Projektmanagement

8.1 Projektorganisation

Das Projektteam besteht aus zwei Mitgliedern, welche für die gesamte Ausführung des Projektes verantwortlich sind.

Organisationsstruktur

Person	E-Mail
Roman Federer (rf)	rfederer@hsr.ch
Cornel Eberle (ce)	ceberle@hsr.ch

Tabelle 11: Projektmitglieder

Projektbetreuung

Person	E-Mail
Prof. Dr. Andreas Steffen	andreas.steffen@hsr.ch
Martin Willi	mwilli@hsr.ch

Tabelle 12: Ansprechpersonen

8.2 Projektplan

Das Frühjahrssemester der Hochschule für Technik Rapperswil HSR führt vom 09.04.2009 bis zum 15.04.2009 eine Woche Ferien. Darum streckt sich der Projektplan über 17 Wochen inklusive der Ferienwoche zwischen den Wochen 8 und 9 (KW 15 und 16).

8.2.1 Arbeitspakete

In den Phasen Analyse und Realisierung werden die Arbeitspakete zu Begin jeweils im Bezug auf eine Single-Sender-Lösung bearbeitet. Dies ist der einfachere Ansatz einer IPsec Multicast Lösung. Am

Ende der beiden Phasen wird dann jeweils das bestehende Konzept, bzw. die bestehende Lösung, auf Multi-Sender-Funktionalität erweitert.

8.2.1.1 Projektmanagement

Projektplanung	
Beschreibung	Projektplan erstellen, Arbeitspakete definieren, Risikomanagement, sowie das stetige Nachführen der Planung während des Projekts.
Zeitspanne	Woche 3 bis 17
Verantwortlichkeit	Roman Federer, Cornel Eberle

Infrastruktur	
Beschreibung	Evaluierung und Aufbau der notwendigen Entwicklungsumgebung zur Durchführung der Analyse und Realisierung des Simulators.
Zeitspanne	Woche 2 bis 4
Verantwortlichkeit	Roman Federer, Cornel Eberle

8.2.1.2 Analyse

Beitritt Multicast-Gruppe	
Beschreibung	In diesem Arbeitspaket werden die Anforderungen an den Beitritt einer IPsec Multicast-Gruppe analysiert. Dazu die folgenden Autorisierungsverfahren genauer betrachtet: • GCKS: Policy Datenbank • X.509 Attributzertifikat • SAML Assertions Ebenfalls geht es auch um das Finden des zentralen Gruppenverwalters: • Der Empfänger findet den Verwalter über den Multicast-Traffic • Der Empfänger kennt die IP des Verwalters
Zeitspanne	Woche 4 bis 5
Verantwortlichkeit	Cornel Eberle

Rekeying	
Beschreibung	Beim Rekeying (Wechsel der Verschlüsselungsinformationen) müssen alle Gruppenmitglieder über die neuen Schlüssel informiert werden. Dabei gibt es einige wichtige Punkte zu beachten. *Zeitpunkt* – Ein Schlüsselwechsel kann periodisch und/oder bei einem Bei- bzw. Austritt aus einer Gruppe erfolgen. *Verteilung* – Die Schlüssel können per Unicast oder Multicast an die Gruppenmitglieder verteilt werden. *Übergangsphase* – Beim Schlüsselwechsel muss darauf Rücksicht genommen werden, dass nicht alle Teilnehmer zur selben Zeit die neuen Schlüssel erhalten. Wie kann ein unterbrechungsfreier Wechsel durchgeführt werden? *Kanal* – Die Schlüsselinformationen können über einen eigenen oder einen bestehenden Kanal (SA – Security Association) übertragen werden.
Zeitspanne	Woche 4 bis 5
Verantwortlichkeit	Roman Federer

Nachrichtendefinition	
Beschreibung	Für den Bei- beziehungsweise Austritt eines Mitglieds müssen Nachrichten versandt werden. Dazu können die INFORMATIONAL-Nachrichten des IKEv2 oder neudefinierte Nachrichten verwendet werden. Die Struktur der Nachrichten soll festgelegt werden.
Zeitspanne	Woche 6
Verantwortlichkeit	Roman Federer

Austritt Multicast-Gruppe	
Beschreibung	Der Austritt aus einer Multicast Gruppe kann entweder explizit erfolgen, in dem das Mitglied über ein Meldung den Austritt bekannt gibt, oder das Mitglied verlässt die Gruppe ohne Meldung.
Zeitspanne	Woche 6
Verantwortlichkeit	Cornel Eberle

Paket Adressierung	
Beschreibung	Damit die Pakete an die richtigen Empfänger gelangen, müssen sie korrekt Adressiert werden. *Traffic Selectors* – Die Quell- und Zieladresse(n) müssen bestimmt werden.
Zeitspanne	Woche 6
Verantwortlichkeit	Roman Federer

Fehlerfälle	
Beschreibung	Es soll auf Spezialfälle der IPsec Multicast Lösung eingegangen werden. Der standardmässige Ablauf ist auf Bad Cases zu untersuchen und für diese Abweichungen sollen Lösungen generiert werden.
Zeitspanne	Woche 7
Verantwortlichkeit	Cornel Eberle

Multi-Sender	
Beschreibung	Mit einer Multi-Sender-Infrastruktur besteht für jedes Gruppenmitglied die Möglichkeit Verkehr zu empfangen und zu generieren. Das bestehende Konzept soll um diesen Ansatz erweitert werden.
Zeitspanne	Woche 7
Verantwortlichkeit	Roman Federer

8.2.1.3 Realisierung

Das erarbeitete Konzept zum Thema *Multicast Security using IPsec* soll in einem High-Level Protokoll-Simulator realisiert werden. Die Voraussetzungen können aus dem Punkt *2 Aufgabenstellung* entnommen werden.

Multicastteilnehmer	
Beschreibung	Als erstes wird in der Phase Realisierung ein multicast-fähiger Netzwerkteilnehmer erstellt. Dieser soll die Grundstruktur für die IPsec Multicast Komponente darstellen und in den weiteren Paketen um die entsprechenden Security-Funktionalitäten erweitert werden.
Zeitspanne	Woche 8
Verantwortlichkeit	Roman Federer, Cornel Eberle

Beitritt Multicast-Gruppe	
Beschreibung	Das Arbeitspaket enthält die Entwicklung verschiedener Anmelde-Vorgänge, wie sie während der Analyse definiert wurden.
Zeitspanne	Woche 8 bis 11
Verantwortlichkeit	Cornel Eberle

Rekeying	
Beschreibung	Das erarbeitete Konzept des Schlüsselwechsels (Rekeying) wird in den Simulator integriert.
Zeitspanne	Woche 8 bis 11
Verantwortlichkeit	Roman Federer

Austritt Multicast-Gruppe	
Beschreibung	Die notwendige Funktionalität für das explizite Abmelden wird implementiert. Ebenfalls wird die Lösung für das nachrichtenlose Verschwinden eines Hosts programmiert. Dies kann wahrscheinlich standardmässig via IKEv2 DELETE Notifications, respektive Timeout der CHILD_SA auf der GCKS-Seite realisiert werden.
Zeitspanne	Woche 10 bis 11
Verantwortlichkeit	Cornel Eberle

Multi-Sender	
Beschreibung	Der High-Level Protokoll-Simulator wird um die Funktionalität des Multi-Senders erweitert.
Zeitspanne	Woche 12 bis 13
Verantwortlichkeit	Roman Federer, Cornel Eberle

TESLA (Optional)	
Beschreibung	Der Verkehr in einer Multicast-Gruppe kann nur durch ein symmetrisches Verfahren authentisiert werden. Das führt dazu, dass innerhalb der Gruppe die Authentisierung nicht gewährleistet ist. Es bestehen bereits Ansätze in der Theorie, wie diese Probleme zu lösen sind. Dazu gehört das TESLA-Verfahren. In diesem Arbeitspaket soll untersucht werden, wie TESLA im Zusammenhang mit IPsec Multicast verwendet werden kann.
Zeitspanne	Woche 14 bis 16
Verantwortlichkeit	Roman Federer, Cornel Eberle

8.2.1.4 Dokumentation

Dokumentation	
Beschreibung	Zu diesem Arbeitspaket gehören sämtliche Dokumentationsarbeiten.
Zeitspanne	Woche 1 bis 17
Verantwortlicher	Roman Federer, Cornel Eberle

Präsentation	
Beschreibung	Vorbereitung und Durchführung der Projekt-Präsentation.
Zeitspanne	Woche 16 bis 17
Verantwortlichkeit	Roman Federer, Cornel Eberle

8.2.1.5 Wiederkehrende Arbeiten

Knowhow	
Beschreibung	Einarbeitung in die bestehenden Technologien. Dazu gehören die folgenden Themengebiete: *IP Multicast* – Verteilt den Netzwerkverkehr an mehrere Empfänger *IKEv2* – Internet Key Exchange Version 2 wird für den Schlüsselaustausch zweier Kommunikationspartner benötigt. *strongSwan* – IPsec-Implementierung für Linux *GKDP* – Group Key Distribution Protocol verteilt die Schlüssel an die Gruppenmitglieder Weitere Informationen wichtiger RFCs (Request for Comments) können aus Abschnitt 2 Aufgabenstellung entnommen werden.
Zeitspanne	Woche 1 bis 17
Verantwortlichkeit	Roman Federer, Cornel Eberle

Reviews und Sitzungen	
Beschreibung	Wöchentliche Reviews sowie ausserordentliche Beratungen mit den Projektbetreuern.
Zeitspanne	Woche 1 bis 17
Verantwortlichkeit	Roman Federer, Cornel Eberle

8.2.2 Meilensteine

Die 16 Arbeitswochen der Bachelorarbeit strecken sich durch die Ferienwoche zwischen den Kalenderwochen 15 und 16 im Frühlingssemester über insgesamt 17 Kalenderwochen.

Inhalt	Ende	Meilenstein
Knowhow Aufbau und Projektplan	W04 / KW 11	MS1: Projektplan
Konzept	W07 / KW 14	MS2: Konzept
Single-Sender Simulator	W11 / KW 18	MS3: Single-Sender
Multi-Sender Funktionalität	W13 / KW 20	MS4: Multi-Sender
TESLA (optional)	W16 / KW 23	MS5: TESLA
Abschluss	W17 / KW 24	MS6: Abschluss

Tabelle 13: Meilensteine

8.3 Risikomanagement

8.3.1 C# Funktionalität

In der Programmiersprache Microsoft .NET C# besitzt das Projektteam im Bezug auf die IP Multicastfunktionalitäten nur geringe bis keine Erfahrungen. Es wird davon ausgegangen, dass die benötigten Funktionen unterstützt werden. Ist dies nicht der Fall wird die Realisierung des Simulators erschwert oder möglicherweise verunmöglicht.

Massnahmen

Die benötigten Funktionalitäten sollen im Voraus durch kleine Softwaretests überprüft werden. Der Zeitpunkt der Tests soll so früh als möglich gewählt werden. Ergeben die Abklärungen, dass der Simulator mit Microsoft .NET C# nicht mit geeignetem Aufwand realisiert werden kann, werden mögliche Alternativen in Betracht gezogen.

Risikoeinstufung

Da es sich bei Microsoft .NET C# um eine Standardprogrammiersprache handelt, wird von einem tiefen Risiko ausgegangen.

Endtermin

Vor dem Beginn der Realisierung des Simulators muss das Risiko beseitigt werden und vorab die nötigen Abklärungen und Tests durchgeführt werden. Der Endtermin wird auf das Ende der Woche 6 gesetzt.

8.3.2 Knowhow

Das Projektteam versucht in kurzer Zeit sich möglichst viel Grundwissen im Zusammenhang mit IPsec und Multicast anzueignen. Eine umfangreiche Kenntnis des Themas sowie entsprechende Erfahrung fehlt. Es wäre möglich, dass entsprechende Konzepte und Entwicklungen aufgrund mangelnden Wissens fehlerhaft sein können.

Massnahmen

Durch die Betreuung von Prof. Dr. Andreas Steffen und Dipl. Ing. Martin Willi stehen ausgesprochen Kompetente Personen zur Verfügung. Das Projektteam soll Lösungen jeweils mit Ihnen besprechen und verifizieren lassen.

Risikoeinstufung

Das Risiko wird als mittelmässig eingestuft.

Endtermin

Für dieses Risiko kann kein Endtermin festgelegt werden. Allfällige Probleme gehören zur Schwierigkeit dieses Projekts und müssen durch Diskussion mit den Betreuern beseitigt werden.

8.3.3 Zeitplanung und -auswertung

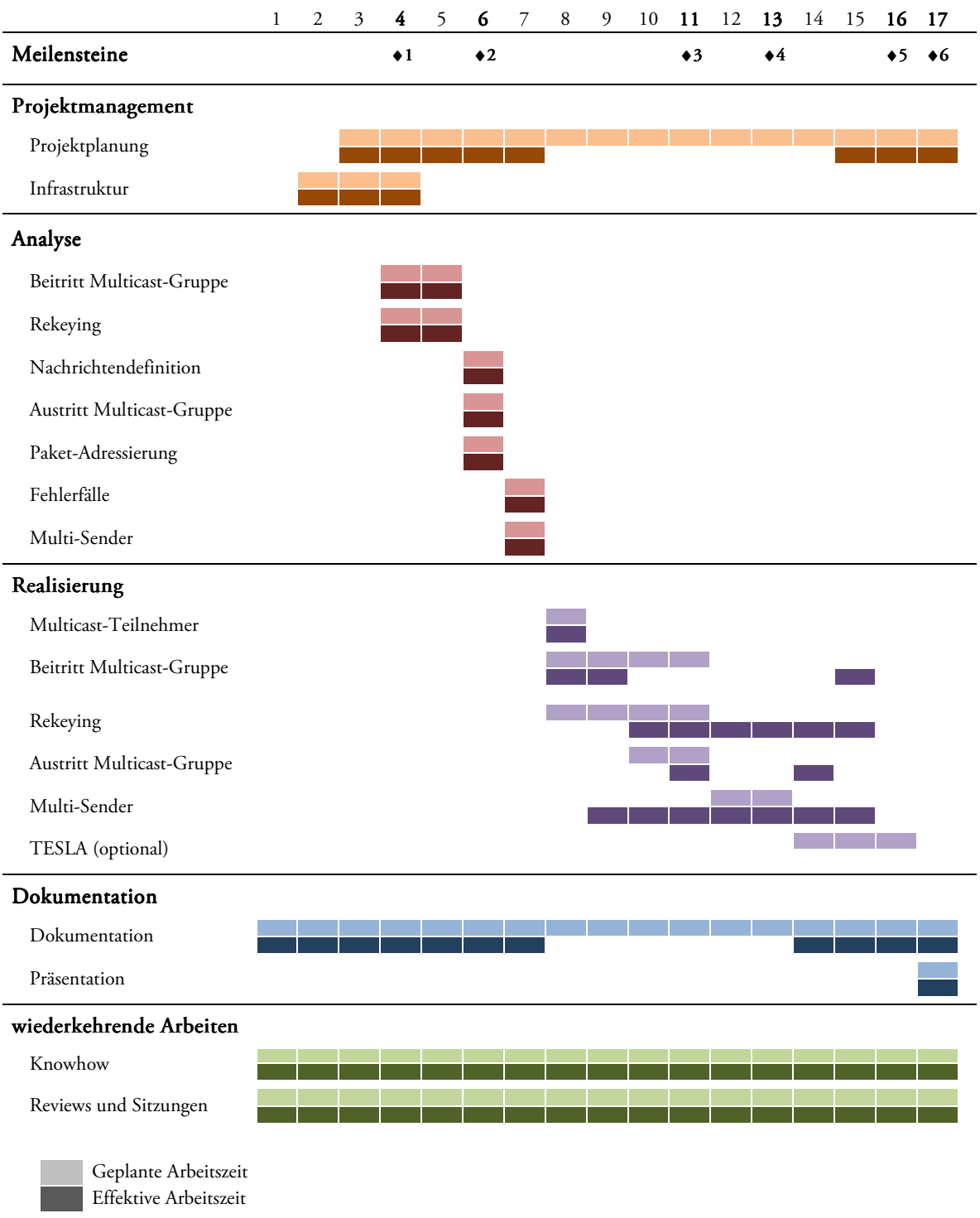

Abbildung 54: Zeitplanung und -auswertung

Die effektive Arbeitsleistung deckt sich grundsätzlich mit dem Zeitplan. Im Bereich der Dokumentation und der Realisierung der Arbeit sind Unterschiede erkennbar.

Die Abweichungen in der Realisierung sind mit dem erhöhten Aufwand für das Rekeying wie auch für den Multi-Sender zu begründen. Da bereits früh mit der Berücksichtigung der Multi-Sender-Architektur begonnen wurde, wurde dieses Arbeitspaket ein wenig vorgezogen.

Die Vernachlässigung der Dokumentation während den Wochen 8 -13 ist mit den vermehrten Änderungen der Architektur des Simulators zu begründen. Eine sofortige Dokumentation der Software wurde aus diesem Grunde nicht als sinnvoll erachtet.

Auf Grund der aufwändigen Realisierung des Simulators wurde auf die optionale Implementierung des TESLA-Verfahrens verzichtet.

8.3.4 Stundenabrechnung

Das folgende Diagramm zeigt den Verlauf des Aufwandes in Stunden pro Woche über die gesamte Projektlaufzeit hinweg. Während den ersten 13 Wochen arbeitete das Projektteam jeweils zwischen 15 und 25 Stunden pro Wochen. Der Mehraufwand am Ende der Arbeit ist eine Folge der aufwändigen Dokumentationsarbeiten.

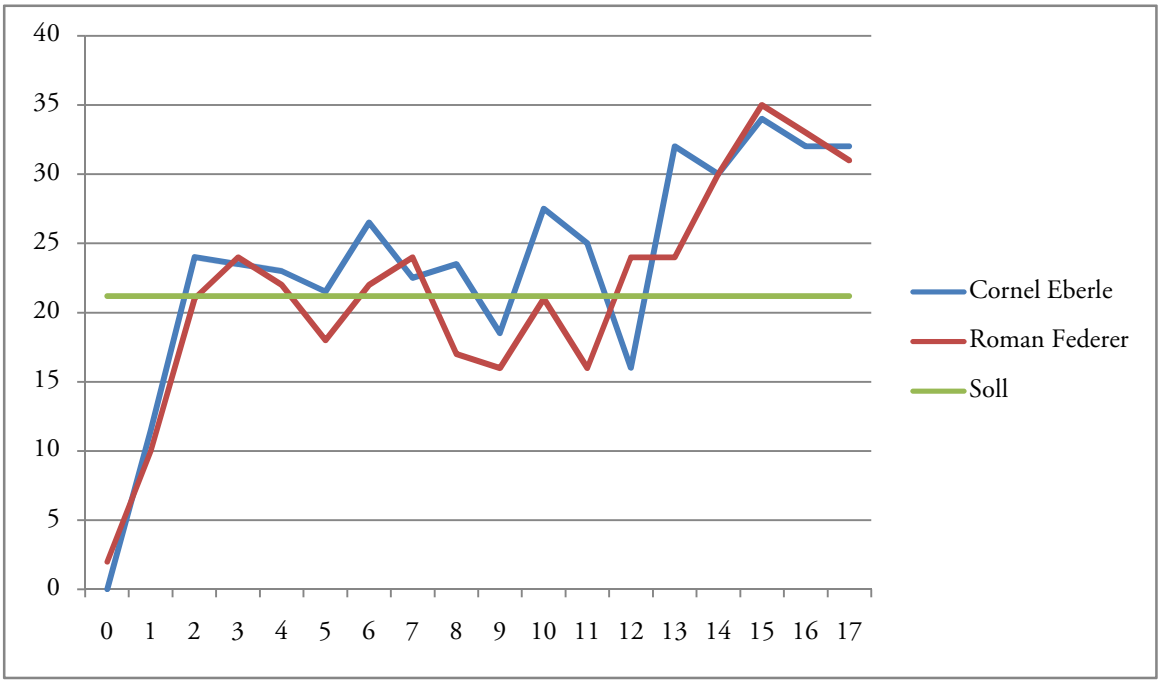

Abbildung 55: Stundenabrechnung

Die Tabelle 14 enthält den Durchschnittswert der geleisteten Stunden pro Woche und das Total der aufgewandten Zeit pro Projektmitglied.

Wer	Ø	Σ
Cornel Eberle	24.8 h	423 h
Roman Federer	22.9 h	390 h
Richtwert Modulbeschreibung	21.7 h	360 h

Tabelle 14: Stundenabrechnungs-Tabelle

9 Erfahrungsberichte

9.1 Cornel Eberle

Kurz bevor wir mit dieser Bachelorarbeit starteten, legten wir die Prüfung zum Modul Internet Sicherheit 1 ab. Unter anderem kam darin auch das Thema IPsec mit IKEv2 vor. Die Anfangsphase dieses Projekts zeigte relativ schnell, dass unsere Vorkenntnisse in diesem Bereich sehr dürftig waren. Dementsprechend waren wir am Anfang sehr intensiv damit beschäftigt, uns die theoretischen Kenntnisse anzueignen.

Während der Entwicklung des Konzepts und dessen Implementierung stieg das Wissen zum Thema recht schnell an. Ebenfalls trug die gute Arbeitsatmosphäre im Team viel zu einem planmässigen Arbeitsfortschritt bei.

Positive Erfahrungen

Bereits bei der Projektausschreibung bekundeten wir unser Interesse an der Verbindung einer Programmieraufgabe mit dem Thema Internet Sicherheit. Auch während dem Projekt empfand ich das Themengebiet sehr interessant und lehrreich. Zudem erlebte ich die Kombination aus der Entwicklung des theoretischen Konzepts und der Realisierung des Simulators als sehr gute Abwechslung.

Während der gesamten Projektdauer war die Zusammenarbeit mit den Betreuern sehr positiv. Vor allem während der konzeptionellen Phase konnten wir viel profitieren. Die Überlegungen im Team und die anschliessende Diskussion mit den Betreuern habe ich sehr geschätzt.

Negative Erfahrungen

Die Implementierung eines Netzwerk-Protokolls bedeutete für mich Programmier-Neuland. Dementsprechend war unsere Arbeit geprägt von vielen Veränderungen in der Struktur. Gewisse Teile der Software wurden mehrfach umgebaut, was viel Zeit in Anspruch nahm und mühsame Arbeit bedeutete. Im Nachhinein wäre es besser gewesen, von Anfang an mehr fremde Hilfe in Anspruch zu nehmen.

Das Durchlesen der verschiedenen RFCs war für mich eine recht beschwerliche Arbeit. Weniger war die Komplexität des Inhalts, sondern viel mehr die mangelnde Struktur, die fehlenden Illustrationen und die vielen Verweise zu weiteren Dokumenten eine anstrengende Angelegenheit.

Ausblick

Mit dieser Bachelorarbeit habe ich sehr viel Hintergrundwissen zu IPsec und IKEv2 erhalten. Ebenfalls habe ich einiges zur Realisierung eines Netzwerk-Protokolls gelernt. Ich denke, dies wird mir in meiner weiteren Berufstätigkeit zu gute kommen. Sicherlich werde ich auch die Weiterentwicklung von IPsec, IKEv2 und strongSwan verfolgen. Und sollte irgendwann einmal ein Standard zu IPsec Multicast auftauchen, werde ich diesen mit besonderem Interesse durchlesen.

9.2 Roman Federer

Nach dem Abschluss der hardwarenahen Studienarbeit zum Thema RFID wollten wir im Rahmen der Bachelorarbeit ein Projekt realisieren, das aus einer Programmieraufgabe bestand. Ebenfalls waren wir darauf bedacht, den Forschungsaspekt nicht ausser Acht zu lassen. Da uns die Internetsicherheit während dem fünften Semester faszinierte, einigten wir uns auf die Arbeitsausschreibung mit dem Titel „Multicast Security using IPsec".

Positive Erfahrungen

Auf die Einarbeitung in die Technologie des IKEv2 und dem Studium bestehender Ansätze, folgte das Entwickeln des theoretischen Konzepts. Dieser Teil der Arbeit war sehr interessant und lehrreich. Während dieser Projektphase mussten wir einige knifflige Probleme lösen. Dabei stellten sich die Reviews mit dem Betreuerteam als hilfreich heraus. Allfällige Fragen konnte wöchentlich geklärt werden, um Projektverzögerungen entgegenzuwirken.

Der zweite Teil der Bachelorarbeit bestand aus der Entwicklung des Simulators. Da das Programmieren wie erwähnt während der Studienarbeit zu kurz kam, machte er mir umso mehr Spass einen High-Level Simulator zu entwickeln.

Äusserst positiv ist die Zusammenarbeit innerhalb des Projektteams zu bewerten. Problemlos liessen sich bei Meinungsverschiedenheiten Kompromisse finden.

Negative Erfahrungen

Nicht als Leseratte bekannt bin, musste ich mich zu Beginn des Projekts überwinden, bestehende Dokumentationen durchzulesen, um mein Knowhow aufzubauen und das Wissen zu erweitern. Die gewöhnungsbedürftige Schreibweise der RFCs erschwerte die Einarbeitung zusätzlich.

Da wir in der Programmierung eines Protokollsimulators keine Erfahrungen hatten, betrieben wir einigen Aufwand bis wir die optimale Architektur gefunden hatten. Die verschiedenen Umbauarbeiten der Software nahmen einige Zeit in Anspruch.

Ausblick

Während der Bachelorarbeit konnte ich mein Wissen im Bereich der Programmierung und Internetsicherheit stark erweitern. Die Erarbeitung eines Konzepts und dessen Realisierung machte mir sehr viel Spass. Für meine spätere berufliche Tätigkeit werde ich einiges aus diesem Projekt mitnehmen können. Den weiteren Werdegang des Konzepts werde interessiert beobachten und gespannt die Entwicklung von strongSwan verfolgen.

10 Verzeichnisse

10.1 Abkürzungsverzeichnis

AH	Der Authentication Header
DPD	Dead Peer Detection
ESP	Encapsulating Security Payload
GCKS	Group Controller and Key Server
GKGP	Group Key Distribution Protocol
GKMA	Group Key Management Algorithm
IGMP	Internet Group Management Protocol
IKE	Internet-Key-Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
KEK	Key Encryption Key
NAT	Network Address Translation
RFC	Request for Comments
SA	Security Association
SPI	Security Parameter Index
TCP	Transmission Control Protocol
TEK	Transport Encryption Key
UDP	User Datagramm Protocol
VPN	Virtual Private Network

10.2 Tabellenverzeichnis

10.3 Abbildungsverzeichnis

10.4 Literaturverzeichnis

1. **Neumann, E.** *Multicast Anwendungen auf Basis IPSec.*

2. **Fenner, W.** RFC 2236 - Internet Group Management Protocol, Version 2. [Online] http://tools.ietf.org/html/rfc2236.

3. **Van Moffaert, A. und Paridaens, O.** Security issues in Internet Group Management Protocol version 3 (IGMPv3). [Online] 2002. http://tools.ietf.org/id/draft-irtf-gsec-igmpv3-security-issues-01.txt.

4. **Baugher, M., et al.** RFC 4046 - Multicast Security (MSEC) Group Key Management Architecture. [Online] http://tools.ietf.org/html/rfc4046.

5. **Dondeti, L., Xiang, J. und Rowles, S.** GKDP: Group Key Distribution Protocol. [Online] http://tools.ietf.org/html/draft-ietf-msec-gkdp-01.

6. **Kaufman, C.** RFC 4306 - Internet Key Exchange (IKEv2) Protocol. [Online] http://tools.ietf.org/html/rfc4306.

7. **Wallner, D., Harder, E. und Agee, R.** RFC 2627 - Key Management for Multicast: Issues and Architectures. [Online] http://tools.ietf.org/html/rfc2627.

8. **Balenson, D., McGrew, D. und Sherman, A.** Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization. [Online] http://tools.ietf.org/html/draft-balenson-groupkeymgmt-oft-00.

9. **Briscoe, Bob.** *MARKS: Zero Side Effect Multicast Key Management using Arbitrarily Revealed Key Sequences.*

10. **Apache.** Apache Logging Services Project. [Online] http://logging.apache.org/.