Multicast-Verschlüsselung und Keymanagement

Matthias Neu
Fachbereich Informatik
Hochschule Bonn-Rhein-Sieg
Email: matthias.neu@smail.inf.h-brs.de

Zusammenfassung—Die Multicast Kommunikation bietet in zahlreichen Situationen große Vorteile gegenüber der Unicast Kommunikation. Soll eine solche Kommunikation jedoch sicher stattfinden, so sollte ein Multicast-Verschlüsselungsverfahren und damit einhergehend ein Multicast Keymanagement Verfahren verwendet werden. Es gibt zahlreiche solcher Verfahren, wobei die Auswahl des jeweiligen Verfahrens von verschiedenen Rahmenbedingungen abhängt. Des Weiteren ist es von großer Bedeutung die Sicherheitsaspekte in diesem Bereich zu betrachten, da die sichere Multicast Kommunikation gegenüber der sicheren Unicast Kommunikation weitaus weniger intensiv erforscht ist und zusätzliche Angriffsvektoren enthält.

1 Einleitung

Die sichere Datenübertragung zwischen zwei Subjekten, welche mittels Unicast miteinander kommunizieren, gilt als sehr gut erforscht und ausgreift, jedoch bekommt auch die sichere Kommunikation in einer Gruppe, also einer Multicast Situation, eine stetig wachsende Bedeutung. Zu solchen Anwendungen gehören beispielsweise Audio- und Video-Konferenzen zwischen mehreren Partnern in der Wirtschaft, der Politik, oder dem Militär, Abonnenten-basiertes Pay-TV und Softwareaktualisierungen. Betrachtet man beispielsweise den Fall einer Videokonferenz zwischen mehreren Parteien, in welcher es einen oder mehrere Sender und viele Empfänger gibt. So ist eine solche Kommunikation mittels Unicast sehr ineffizient, da der Sender das gleiche Signal an jeden Empfänger einzeln schicken müsste, was bei einer großen Anzahl an Empfängern in einer hohen zu sendenden Datenmenge resultieren würde. Auch müssten für eine sichere Übertragung alle Daten für jeden Empfänger separat verschlüsselt werden, was in einem hohen Aufwand für den Sender resultieren würde. Deutlich effizienter wäre die Übertragung mittels Multicast, da das Signal in diesem Fall lediglich einmal verschickt werden müsste. Hätten alle Mitglieder der Gruppe den gleichen geheimen Schlüssel (Gruppenschlüssel), sodass das Signal mit diesem mittels eines symmetrischen Verschlüsselungsverfahrens verschlüsselt und per Multicast versendet werden könnte, würde sich der Aufwand für den Sender minimieren. Die Integrität der versendeten Daten ist gegeben, da unter der Annahme, dass das verwendete Verschlüsselungsverfahren sicher ist, die verschlüsselten Informationen lediglich mit dem Gruppenschlüssel entschlüsselt werden könnten [1], [2].

Eine besonders wichtige Rolle im Bereich der Multicast-Verschlüsselung stellt das Keymanagement dar, da im Gegensatz zu dem Unicast-Fall das Kommunikationsumfeld nicht statisch, sondern dynamisch ist. Grade in größeren Gruppen kann es häufig vorkommen, dass Gruppenmitglieder die Gruppe verlassen, oder neue Mitglieder hinzukommen. Hierbei sind die Sicherheitseigenschaften der Forward secrecy und der Backward Secrecy von großer Bedeutung.

Forward secrecy ist dann gegeben, wenn ein ehemaliges Mitglied einer Gruppe nicht mehr Informationen aus der verschlüsselten Gruppen-Nachricht gewinnen kann, als ein beliebiges Subjekt, welches noch nie in der Gruppe war. Verlässt beispielsweise ein Geschäftspartner die Videokonferenz, so soll es ihm nicht möglich sein diese weiterhin zu sehen, indem er die von den Sendern ausgehende Kommunikation abhört.
Backward Secrecy ist dann gegeben, wenn es einem neuen Gruppenmitglied nicht möglich ist, bereits vor seinem Eintritt in die Gruppe stattgefundene Kommunikation zu entschlüsseln. Wird beispielsweise ein neuer Geschäftspartner in eine Videokonferenz hinzugefügt, so soll es ihm nicht möglich sein alte Abschnitte der Konferenz zu sehen, selbst wenn er im Vorhinein die Kommunikation der Sender belauscht hat [1], [2].

Um diese Sicherheitseigenschaften zu gewährleisten ist es notwendig den Gruppenschlüssel beim Eintritt und beim Verlassen eines Gruppenmitgliedes zu erneuern, wobei vor allem im Fall des Verlassens sichergestellt sein muss, dass der neue Gruppenschlüssel nicht vom alten abhängt. Für ein solches Keymanagement wurden in den letzten Jahrzehnten zahlreiche Protokolle entwickelt, welche sich grob in drei Klassen einteilen lassen:
Zentralisiert: In einem zentralisierten Keymanagement Protokoll gibt es eine zentrale Einheit (Schlüsselserver), welche die Gruppe kontrolliert und für die Schlüsselverteilung zuständig ist. Viele Protokolle dieser Klasse sind vergleichsweise intuitiv und leicht zu verstehen, haben jedoch den Nachteil, dass der Schlüsselserver zu einem limitierenden Faktor werden kann.
Dezentralisiert: Protokolle der dezentralisierten Klasse unterscheiden sich von Protokollen der zentralisierten Klasse dahingehend, dass in dieser mehrere kontrollierende Instanzen existieren, welche das Keymanagement untereinander aufteilen und somit die einzelnen Schlüsselserver entlastet werden.
Verteilt: In einem verteilten Keymanagement Protokoll gibt es keinen Schlüsselserver. Vielmehr wird der Gruppenschlüssel durch die einzelnen Gruppenmitglieder

erstellt, wobei es protokollabhängig ist, ob nur ein Gruppenmitglied Informationen für den geheimen Schlüssel bereitstellt, oder alle Mitglieder der Gruppe zu der Erstellung beitragen [2], [3].

Wichtige Faktoren zur Bewertung von Keymanagement Protokollen ist der Berechnungsaufwand und der Speicherplatzbedarf des Schlüsselservers und der Gruppenmitglieder, die benötigte Bandbreite bei der Kommunikation und der Schlüsselerneuerung. Sehr wichtig ist die Skalierbarkeit im Bezug auf die Gruppengröße. Im Bezug zu der Fragestellung, wann der Gruppenschlüssel aktualisiert werden soll, wird zwischen einem Zeit-basierten und einem Mitglied-basierten Verfahren unterschieden. Ein Zeit-basiertes Verfahren wird dadurch charakterisiert, dass die Schlüsselerneuerung in definierten Zeitintervallen stattfindet, was den Vorteil hat, dass der Aufwand für das Keymanagement nicht zu hoch wird. Allerdings ist bei dieser Methode keine Forward secrecy oder Backward Secrecy gegeben. Im Gegensatz dazu wird bei dem Mitglied-basierten Verfahren der Schlüssel immer dann aktualisiert, wenn ein Mitglied in die Gruppe hinzugefügt oder aus dieser entfernt wird [2].

2 Technische Grundlagen

2.1 Multicast Kommunikation

In der Kommunikation mittels Multicast sendet ein Sender eine Nachricht an eine Gruppe von Empfängern, wobei hierzu lediglich eine Sende-Operation getätigt werden muss, anstatt an jedes Gruppenmitglied eine separate Nachricht schicken zu müssen. Dadurch wird es ermöglicht deutlich effizienter in einer Gruppe kommunizieren zu können, da zum einen der Sender, dadurch das dieser lediglich eine Nachricht senden muss, deutlich entlastet wird und zum anderen das Netzwerk einen geringeren Bandbreitenverbrauch aufweist.
Beispielsweise benötigt das Versenden einer 300 MB großen Nachricht bei einem Ethernet mit einer Geschwindigkeit von 10Mbps (Mega bits per second) ca. 67.7 Stunden bei einer separaten Übertragung per Unicast an jeden einzelnen Empfänger, wohingegen eine solche Kommunikation mittels Multicast in ca. 4 Minuten getätigt werden kann [4].

2.1.1 IP Multicast

Eine in der Praxis relevante Realisierung für eine Multicast Kommunikation stellt IP-Multicast dar. Hierbei wird eine Multicast Gruppe durch eine IPv4 Adresse der Klasse D (der Adressraum liegt zwischen 224.0.0.0 und 239.255.255.255) repräsentiert. Wird an eine solche Adresse eine Nachricht gesendet, so wird eine Kopie dieser Nachricht an alle Mitglieder dieser Gruppe, also an alle Subjekte, welche mit dieser Gruppe assoziiert werden, geschickt. Um eine solche Multicast Gruppe zu organisieren, kann beispielsweise das Internet Group Management Protocol (IGMP) verwendet werden. Mit der Hilfe dieses Protokolls kann jedes beliebige Subjekt jeder beliebigen Multicast Gruppe beitreten, oder diese verlassen. Zur effizienten Koordinierung, an welche Router ein Router eine Multicast Nachricht weiterschicken soll, wird zusätzlich ein Multicast-Routing-Protokoll benötigt.

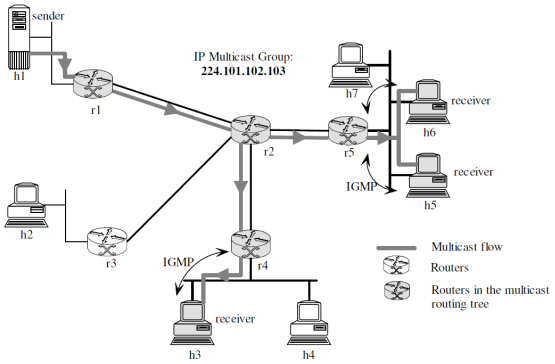

Abbildung 1. Skizzierung einer Multicast Kommunikation [4]

Abbildung 1 skizziert eine Multicast Kommunikation mittels IP-Multicast. Der Sender, welcher kein Mitglied der Multicast Gruppe ist, sendet ein Paket an eine Multicast-Gruppe, welche eindeutig durch ihre Klasse D IP Adresse 224.101.102.103 adressiert wird. Dabei wird die Nachricht zunächst vom Subjekt an den Router r1 gesendet, welcher die Nachricht an den Router r2 weiterleitet, wovon aus die Nachricht kopiert und an die Router r4 und r5 weitergeleitet wird, da hinter diesen Routern Mitglieder der Multicast Gruppe liegen, an welche die Nachricht vom jeweiligen Router weitergeleitet wird [4].

2.2 Multicast Keymanagement

In Fällen, in denen eine Kommunikation in einer Gruppe erwünscht ist, diese Kommunikation jedoch vertraulich und integer sein soll und eine Kontrolle über die Mitgliederschaft einer Gruppe notwendig ist, reicht ein einfaches Multicast Protokoll nicht aus, da die Kommunikation hierbei abgehört werden könnte und Multicast-Routing-Protokolle dahingehend entworfen wurden, Subjekten einer Multicast Gruppe die gesendeten Pakete zukommen zulassen, jedoch nicht dazu, um sicherzustellen, dass keine anderen Subjekte die gesendeten Daten erhalten können. Eine Möglichkeit für dieses Problem ist die Multicast-Verschlüsselung, in welcher nur jedes autorisierte Subjekt den geheimen Gruppenschlüssel besitzt und somit an der Gruppenkommunikation teilnehmen kann. Alle Mitglieder dieser Gruppe sind Mitglieder einer zugrundeliegenden Multicast Gruppe (beispielsweise IP Multicast), jedoch müssen nicht notwendigerweise alle Mitglieder der Multicast Gruppe auch Mitglieder der „sicheren“Gruppe sein. Ein solches Subjekt könnte zwar die verschlüsselte Kommunikation erhalten, wäre allerdings nicht in der Lage diese zu entschlüsseln. Ziel des Keymanagements ist es den Gruppenschlüssel sicher und möglichst effizient an an die Gruppenmitglieder zu verteilen. Ein solches Keymanagement Protokoll soll unabhängig vom darunterliegenden Multicast Protokoll und Multicast-Routing-Protokoll sein [4], [5].

2.3 Authentifiziertes Unicast Keymanagement

Eine äußerst wichtige Voraussetzung für die Sicherheit der Multicast Keymanagement Protokolle ist der Aufbau einer sicheren Unicastverbindung zwischen einem möglichen

neuen Mitglied einer Gruppe und einem Subjekt der Gruppe, welches mit dem potentiell neuen Mitglied die notwendige Kommunikation durchführt, um dieses in die Gruppe zu integrieren.
Eine solche sichere Unicastverbindung kann durch ein authentifiziertes Keymanagement durchgeführt werden. Zwei mögliche Ansätze für ein solches Keymanagement liegen zum einen in der Verwendung einer Public Key Infrastruktur (PKI), also unter der Zuhilfenahme einer dritten, vertrauenswürdigen Entität und zum anderen auf Basis eines geteilten Geheimnisses zwischen den beiden Entitäten, welche einen Schlüssel zur gemeinsamen Kommunikation austauschen wollen.
Das Encrypted Key Exchange [6] (EKE) Protokoll reiht sich in die zweite Kategorie ein, also jener, in welcher die beteiligten Entitäten ein Geheimnis, in diesem Fall ein Passwort, teilen. Ziel ist es, gemeinsam einen Schlüssel zu generieren und dabei zu überprüfen, ob die jeweils andere Entität im Besitz des geheimen Passwortes ist.
Das Protokoll:
Die Entitäten A und B kennen beide das Passwort P und sollen auf Grundlage dessen einen geheimen Schlüssel zur Kommunikation untereinander erstellen.

Vorbereitung: α und β seien öffentliche Parameter, wobei α ein erzeugendes Element von $\mathbb{Z}_\beta^*$ sei.

1) A generiert eine zufällige Zahl $R_A \in \mathbb{Z}_\beta^*$ und berechnet $E_{P'}\left(\alpha^{R_A}\ (mod\ \beta)\right)$, wobei $E_{P'}(m)$ eine symmetrische Verschlüsselungsfunktion und P' ein vom Passwort P abgeleiteter (beispielsweise P als Binärzahlen darstellen und auf die benötigte Länge mit 0'en füllen) Schlüssel ist, mit welchen die Nachricht m verschlüsselt wird.
A schickt $E_{P'}\left(\alpha^{R_A}\ (mod\ \beta)\right)$ und die zu A zugehörige Identität ID_A an B.

2) B generiert eine zufällige Zahl $R_B \in \mathbb{Z}_\beta^*$, entschlüsselt die erhaltene Nachricht $E_{P'}\left(\alpha^{R_A}\ (mod\ \beta)\right)$ mit Hilfe des Passwortes P und berechnet daraufhin $\left(\alpha^{R_A}\right)^{R_B} \equiv \alpha^{R_A * R_B}\ (mod\ \beta)$ und generiert aus diesen Wert den geheimen Schlüssel K. Daraufhin generiert B eine zufällige Zahl $challenge_B \in \mathbb{Z}_\beta^*$.
B schickt $E_{P'}\left(\alpha^{R_B}\ (mod\ \beta)\right)$ und $E_K\left(challenge_B\right)$ an A.

3) A entschlüsselt $E_{P'}\left(\alpha^{R_B}\ (mod\ \beta)\right)$ mit Hilfe des Passwortes P, berechnet daraufhin $\left(\alpha^{R_A}\right)^{R_B}\ (mod\ \beta)$ und generiert daraus den geheimen Schlüssel K. Mit Hilfe von K kann A nun

$$E_K\left(challenge_B\right)$$

entschlüsseln und so an den Wert $challenge_B$ gelangen. Daraufhin generiert A eine zufällige Zahl $challenge_A \in \mathbb{Z}_\beta^*$. A schickt $E_K\left(challenge_A || challenge_B\right)$ an B, wobei $||$ für die Konkatenation steht.

4) B entschlüsselt $E_K\left(challenge_A || challenge_B\right)$ und überprüft, ob die zuvor generierte Zufallszahl $challenge_B$ einem Suffix der entschlüsselten Nachricht entspricht. Wenn ja:
B sendet $E_K\left(challenge_A\right)$ an A.

5) A entschlüsselt die erhaltene Nachricht mit dem Schlüssel K und überprüft, ob die entschlüsselte Nachricht der zuvor erstellten Zufallszahl $challenge_A$ entspricht.

Der im Protokoll generierte Schlüssel K stellt den ausgetauschten Schlüssel zur Unicast Kommunikation dar. In Schritt 4) überprüft B, ob A $challenge_B$ und somit auch das geheime Passwort P kennt, wohingegen A im Schritt 5) überprüft, ob B das Passwort P kennt. Das Passwort P trägt nicht zur Sicherheit des ausgetauschten Schlüssels bei, sondern dient zur Authentifizierung der beteiligten Parteien. Würde nun also eine dritte Entität die Kommunikation zwischen A und B belauschen und im Nachhinein das geheime Passwort erfahren, so würde dieses nicht helfen, um den ausgetauschten Schlüssel zu berechnen. Versucht eine unberechtigte Entität C sich als A auszugeben und einen Schlüssel mit B auszutauschen, so müsste C das Passwort P dabei erraten. Ein Schlüsselaustauschversuch entspricht somit einem erratenen Passwort. Damit es C nicht möglich ist das Passwort P durch einen Brute-Force Angriff zu erraten, sollte B Gegenmaßnahmen gegen einen solchen Angriff implementieren. Eine mögliche Gegenmaßnahme wäre es die Anzahl der Schlüsselaustauschversuche pro Zeitintervall zu beschränken, oder gar eine Wartezeit nach einem Fehlversuch einzuführen, dessen Dauer von der vorangegangen Anzahl an Fehlversuchen abhängt [6].

3 Zentralisierte Protokolle Keymanagement Protokolle

3.1 Logical Key Hierarchy Protokoll

Das Logical Key Hierachy Protokoll [7], [8] (LKH) stellt ein zentralisiertes Keymanangement Protokoll dar, welches auf einer (in der Regel binären) Baumstruktur basiert, welche vom Schlüsselserver kontrolliert wird. Jeder Knoten des Baumes enthält einen Schlüssel-Verschlüsselungs-Schlüssel (KEK, key encryption key), wobei der KEK im Wurzelknoten den Gruppenschlüssel darstellt und jedes Blatt des Baumes einem User zugeordnet ist. Jeder User kennt den KEK des ihm zugeordneten Blatt-Knotens und jene KEKs, welche jenen Knoten zugeordnet sind, die auf dem Pfad des Blatt-Knotens zu dem Wurzelknoten (inklusive des Wurzelknotens) liegen. Abbildung 2 skizziert einen solchen Baum

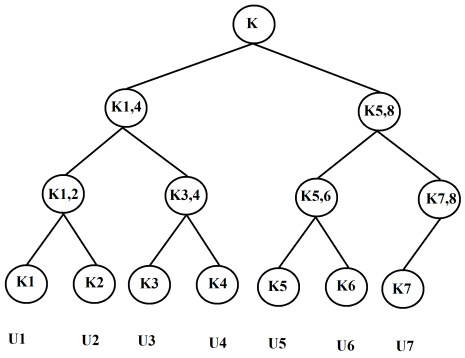

Abbildung 2. Skizzierung eines LKH Baumes mit sieben Usern

für sieben User. In diesem Beispiel kennt beispielsweise der

User „U4“ die KEKs „K4“, „K3,4“, „K1,4“ und „K“, wobei Letzterer der Gruppenschlüssel ist.
Tritt ein neuer User der Gruppe bei, so muss ein neuer Blatt-Knoten und ein zugehöriger Schlüssel für jenen User erzeugt und per Unicast an diesen gesendet werden werden. Daraufhin werden alle Schlüssel, welche auf dem Pfad des Blatt-Knotens zu dem Wurzelknoten liegen erneuert, um die Backward Secrecy zu gewährleisten. Diese neuen Schlüssel werden in einer Multicast Nachricht verschickt, wobei jeder Schlüssel zweimal in verschlüsselter Form, jeweils verschlüsselt mit dem Schlüssel eines Kinder-Knotens, in dieser Nachricht enthalten ist. Abbildung 3 skizziert den Prozess,

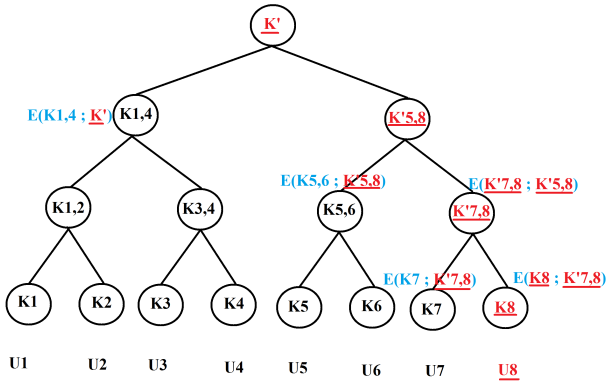

Abbildung 3. Skizzierung des „Hinzufüge“-Prozesses
Die roten Schlüssel signalisieren neu generierte Schlüssel.
Die blauen Verschlüsselungen E(K, m) (k ist der Schlüssel, mit dem die Nachricht m verschlüsselt wird) an einem Knoten signalisieren, dass dieser Teil in der verschlüsselten Multicast Nachricht für diesen Teilbaum relevant ist

in welchem der User „U8“ in die Gruppe hinzugefügt wird. Zunächst erhält dieser von dem Schlüsselserver über einen sicheren Unicast Kanal den Schlüssel „K8“, woraufhin die Schlüssel „K“, „K5,8“ und „K7,8“ erneuert werden müssen. Für die User „U1-4“ ist lediglich der erneuerte Gruppenschlüssel „K“ relevant, welcher in verschlüsselter Form, mit dem Schlüssel „K1,4“ verschlüsselt, der von dem Schlüsselserver versendeten Multicast Nachricht beiliegt. Der User „U7“ hingegen muss alle seiner gespeicherten Schlüssel (exklusive des Schlüssels „K7“) erneuert, wobei zunächst der neue Schlüssel „K’7,8“ mit dem eigenen Schlüssel „K7“ entschlüsselt werden muss, bevor der nächste Schlüssel auf dem Pfad mit dem zuletzt entschlüsselten Schlüssel entschlüsselt werden kann.
Eine weiter Operation in der Gruppe stellt das Verlassen dieser dar. Dies kann auf Wunsch des Users selber geschehen, oder durch eine autorisierte Instanz veranlasst werden. Nachdem ein User die Gruppe verlassen hat, darf es diesem nicht mehr möglich sein die aktuelle Kommunikation in der Gruppe zu entschlüsseln. Daher müssen, wenn ein User die Gruppe verlässt, um die Forward Secrecy zu gewährleisten, alle von ihm bekannten Schlüssel erneuert werden, wobei jener Erneuerungsprozess analog zu dem im Vorhinein beschriebenen durchgeführt wird, wobei der Elternschlüssel des Knotens des entfernten Users nicht mit dem Schlüssel ebenjenen Users verschlüsselt wird. Würde beispielsweise der User „U8“ in dem durch die Abbildung 3 skizzierten Beispiel der Gruppe nicht beitreten, sondern diese verlassen, so würde die vom Schlüsselserver versendete Multicast Nachricht nicht den mit dem Schlüssel „K8“ verschlüsselten Schlüssel „K’7,8“ beinhalten, wodurch es dem User auch nicht möglich wäre die anderen erneuerten Schlüssel zu erhalten, was darin resultiert, dass kein KEK des Baumes dem User bekannt wäre.
Eine triviale Erweiterung dieses Protokolls stellt das LKH+ [9] Protokoll dar, welches sich dahingehend unterscheidet, dass bei dem Hinzufügen eines neuen Nutzers alle betroffenen Schlüssel (jene, welche beim LKH Protokoll erneuert werden) durch die User selber berechnet werden, indem definiert wird, dass jeder neue Schlüssel der Hashwert des alten Schlüssels ist [2].

3.2 One-way Function Tree Protokoll

Ein weiteres zentralisiertes und auf einer (binären) Baumstruktur basierendes Protokoll stellt das One-way Function Tree [10] (OFT) Protokoll dar. Analog zu dem LKH Protokoll stellt in diesem jeder Knoten einen KEK dar und jedes Blatt des Baumes wird einem User zugeordnet. Der zentrale Unterschied liegt in der Aktualisierung der Schlüssel beim Hinzufügen/Entfernen eines Users der Gruppe. Hierbei werden die Schlüssel nicht wie beim LKH Protokoll ausschließlich vom Schlüsselserver zufällig generiert, sondern lediglich die Schlüssel der Blätter eines Baumes, wobei sich die weiteren Schlüssel des Baumes durch die Schlüssel der Blätter fest definiert sind. Ein Schlüssel eines Knotens wird nach Formel 1 durch die Schlüssel der Kinderknoten berechnet, wobei die Funktion h() eine Einwegfunktion (Hashfunktion) darstellt und die Funktion m() für eine Mixingfunktion steht, also eine Funktion, welche aufgrund von zwei Hashwerten deterministisch einen neuen Schlüssel berechnet. Wird als Hashfunktion beispielsweise SHA-256 verwendet, dessen Resultat 256 Bit lang ist und als Verschlüsselungsfunktion der AES-256, dessen Schlüssellänge 256 Bit beträgt, so könnte als Mixingfunktion die XOR Funktion verwendet werden.

$$k = m(h(k_{Kind1}), h(k_{Kind2})) \tag{1}$$

Jeder Knoten des Baumes kann mit zwei Werten assoziiert werden. Zum einen der zugehörige Schlüssel-Wert und zum anderen der geblindete Schlüssel-Wert, also der Hashwert des Schlüssels.
Die Sicherheit des Verfahrens beruht darauf, dass jeder Teilnehmer lediglich die Schlüsselwerte jener Knoten kennt, welche auf dem Weg von dem zu dem User zugehörigen Knoten zum Wurzelknoten liegt und zusätzlich die geblindeten Schlüssel jener Knoten, welche Zwillinge der Knoten sind, welche auf dem beschrieben Weg liegen. Figur 4 skizziert einen Schlüsselbaum mit 7 Usern aus Sicht des Users „U4“. Aus der Skizze wird auch deutlich, dass es für einen User lediglich nötig ist den Blatt-Schlüssel und die geblindeten Schlüssel zu speichern, da aus diesen Informationen die weiteren Schlüssel berechnet werden können.
Wird ein User U der Gruppe hinzugefügt, so wird ein bestehender User U’, dessen zugehöriger Knoten x einen minimalen Abstand zum Wurzelknoten hat, ausgewählt und an den Knoten x werden zwei Knoten x_{link} und x_{rechts} hinzugefügt, wobei der bestehende User U’ mit dem Knoten x_{link} assoziiert wird und der neue User U mit dem Knoten

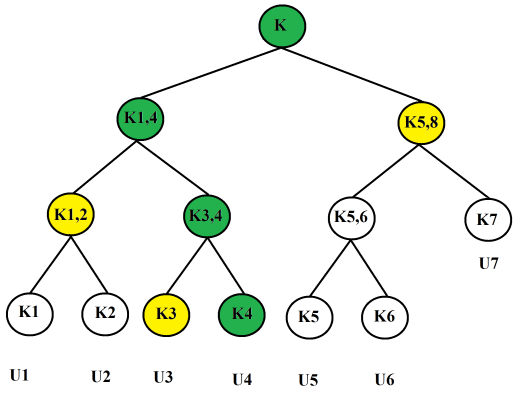

Abbildung 4. Skizze des OTF-Schlüsselbaumes aus Sicht von U4. Die Schlüssel der grün markierten Knoten und die geblindeten Schlüssel der gelben Knoten sind U4 bekannt

x_{rechts}. Daraufhin wird eine Multicast Nachricht verschickt, welche jeden neuen Schlüssel in geblindeter Form und verschlüsselt mit dem Schlüssel des jeweiligen Geschwisterknotens enthält. Zusätzlich muss dem alten User U' sein neuer Schlüssel mitgeteilt werden und dem neuen User U muss über eine sichere Unicast Verbindung der mit ihm assoziierte Schlüssel und die für ihm bestimmten geblindeten Schlüssel mitgeteilt werden, aus welchen dieser alle Schlüssel auf dem Weg zum Wurzelknoten nach Formel 1 berechnen kann. Abbildung 5 skizziert einen solchen

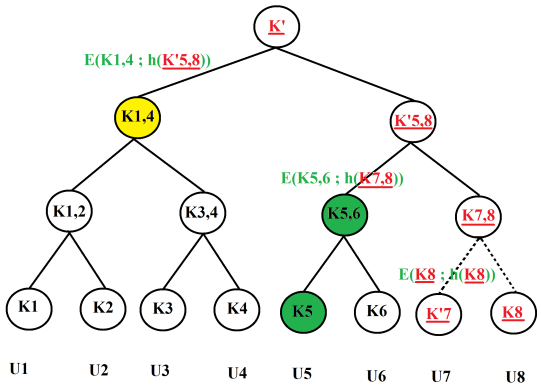

Abbildung 5. Skizze des OTF-Schlüsselbaumes in welchen der User U8 neu hinzugefügt worden ist aus Sicht von User U5. Außerdem wird signalisiert, welche für welchen Teil des Baumes die jeweiligen Teile der Multicast Nachricht relevant sind

Prozess aus der Sicht von User U5. Zuvor hatte der User U7 den kürzesten Weg zum Wurzelknoten, weshalb der zugehörige Knoten aufgeteilt wurde, um den neuen User U8 in den Schlüsselbaum zu integrieren. Außerdem wird für den User „U5“ gezeigt, welche Schlüssel dieser zunächst kennt („K5“,„K5,6“ und „h(K1,4)“). Mit Hilfe des Schlüssels „K5,6“ und dem Teil „E(K5,6 ; h(K'7,8))“ der Multicast Nachricht kann der U5 den geblindeten Schlüssel h(K7,8) berechnen, woraus mit den bekannten Informationen die Schlüssel „K'5,8“ und daraufhin „K'“ berechnet werden können.
Verlässt ein User die Gruppe, so muss der Gruppenschlüssel verändert werden, um die Forward secrecy zu gewährleisten. Dies wird dadurch erreicht, dass der betroffene User aus der Baumstruktur entfernt wird, der zum Geschwisterknoten zugehörige User einen neuen Schlüssel bekommt und dem Elternknoten zugeordnet wird. Daraufhin werden mittels einer Multicast Nachricht, analog des Verfahrens bei dem Hinzufügen eines neuen Users, alle für die jeweiligen User relevanten Informationen für die Berechnung des Gruppenschlüssels bereitgestellt [2], [11], [10].

3.2.1 Kollisionsangriffe auf das OTF-Protokoll

Das OTF Protokoll bietet keine völlige Forward secrecy oder Backward secrecy, da es anfällig gegenüber Kollisionsangriffen ist, also jenen Angriffen, bei denen sich mehrere (eventuell auch bereits aus der Gruppe ausgetretene) User zusammenschließen, um den Gruppenschlüssel für einen Zeitpunkt zu berechnen, an welchen keiner dieser User Kenntnis von diesem haben dürfte [11].

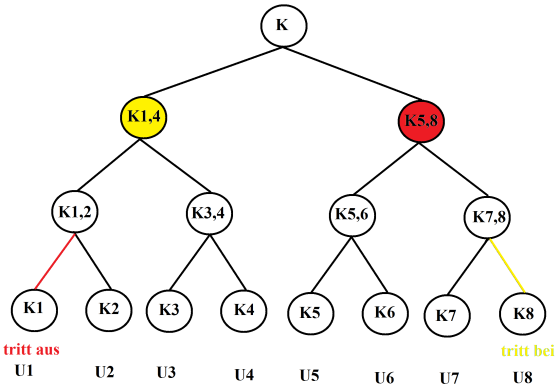

Abbildung 6. Skizze Horng's Angriff

3.2.1.1 Horng's Angriff [12]: Abbildung 6 skizziert die Grundidee hinter dem Kollisionsangriff, bei welchen davon ausgegangen wird, dass ein User (U1) die Gruppe verlässt, daraufhin in der Gruppe Kommunikation stattfindet, welche von U1 eventuell abgehört, aber nicht entschlüsselt werden kann und daraufhin ein User (U8) der Gruppe beitritt. U1 und U8 ist es einzeln nicht möglich die im Vorhinein beschriebene Kommunikation zu entschlüsseln. Schließen sich diese zusammen, so ist es ihnen jedoch sehr wohl möglich. Beim Austritt von U1 wird unter anderem der Schlüssel „K1,4“ erzeugt, weshalb es U1 nicht möglich ist den Gruppenschlüssel K zu berechnen, jedoch bleibt der geblindete Wert „h(K5,8)“, welcher U1 bekannt ist, bestehen. Tritt U8 der Gruppe bei, so wird der Schlüssel „K5,8“ erneuert, wodurch es U8 nicht möglich ist K zu berechnen. Jedoch wird dabei der Schlüssel „K1,4“ nicht verändert und der geblindete Wert dieses Schlüssels wird U8 mitgeteilt. Da beide zusammen „h(K1,4)“ und „h(K5,8)“ kennen, ist es den beiden somit zusammen möglich nach Formel (1) den Gruppenschlüssel K zu berechnen. Aus Sicht von U1 ist somit die Forward secrecy und aus Sicht von U8 die Backward secrecy gebrochen worden [11].

3.2.1.2 Ku und Chen's Angriff [10]: In Horng's Angriff wurde angenommen, dass die zwei zusammenarbeitende User zu zwei unterschiedlichen Teilbäumen gehören

und das zwischen dem Verlassen des ersten Users und dem Hinzukommen des zweiten Users keine Änderung der Gruppe stattgefunden hat. Diese Bedingungen sind jedoch keinesfalls notwendig, damit ein Kollisionsangriff auf das Protokoll funktionieren kann, wie der folgende Angriff zeigt. Figur 7 zeigt den Fall auf, in dem U1 zum Zeitpunkt

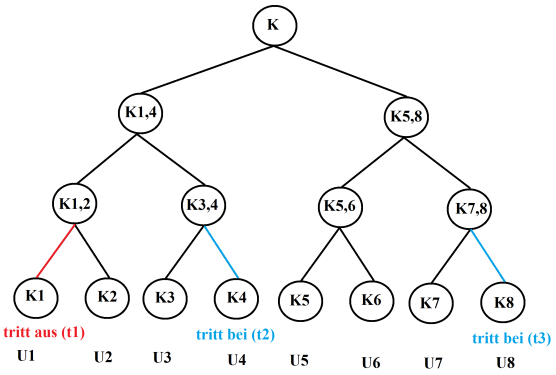

Abbildung 7. Skizze Ku und Chen's Angriff

t1 die Gruppe verlässt, U4 zum Zeitpunkt t2 und U8 zum Zeitpunkt t3 der Gruppe beitreten.
Das zwei User nicht in unterschiedlichen Teilbäumen liegen müssen wird durch folgenden Angriff gezeigt: U1 und U4 können nach Horng's Angriff den Schlüssel „h(K1,4)"(welcher im Zeitraum t1 bis t2 gültig ist) berechnen. Da beide User den geblindeten Wert „h(K5,8)" kennen, ist es beiden Usern möglich den Gruppenschlüssel zu berechnen, welcher im Zeitraum t1 bis t2 gültig war.
Das zwischen dem Verlassen des einen Users und dem Hinzufügen des zweiten Users sehr wohl eine Veränderung der Gruppe stattfinden darf und ein Kollisionsangriff dennoch anwendbar sein kann zeigt der folgende Angriff: Arbeiten U1 und U8 zusammen, so ist es ihnen möglich den Gruppenschlüssel zu berechnen, welcher zum Zeitpunkt t2 bis t3 gültig war, da U1 den in dem Zeitraum gültigen geblindeten Wert von „h(K5,8)" und U8 den geblindeten Wert „h(K1,4)" kennt [11].

3.3 Centralized Flat Key Management Protokoll

Das Centralized Flat Key Management (CFKM) Protokoll ist ein zentralisiertes Keymanagement Protokoll, welches im Gegensatz zu den LKH und OFT Protokollen nicht auf einer Baumstruktur basiert. Im Rahmen des Protokolls wird jedem Nutzer eine eindeutige ID (beispielsweise die IP-Adresse und Port-Nummer) der Länge W (in binärer Form) zugeordnet. Der Schlüsselserver verwaltet eine Tabelle mit W Zeilen, wobei jede Zeilen zwei Einträge enthält und jeder dieser Einträge einen Schlüssel, den Key Encryption Keys (KEKs), darstellt, mit welchen im weiteren Verlauf des Verfahrens der Gruppenschlüssel verschlüsselt zu den Usern transportiert wird. Jeder User besitzt genau W Schlüssel aus der Schlüssel-Tabelle, und zwar jeweils einen Schlüssel aus jeder Zeile der Tabelle. Ist der j'te Wert der ID eine 0, so kennt der User den ersten Schlüssel der j'ten Zeile in der Tabelle, ansonsten kennt er den zweiten Eintrag dieser. Außerdem kennt jeder den Gruppenschlüssel, mit dem die verschlüsselte Kommunikation durchgeführt wird.

TEK 1,1	TEK 1,2
TEK 2,1	TEK 2,2
TEK 3,1	TEK 3,2
TEK 4,1	TEK 4,2

User ID: 1001

Abbildung 8. Skizze CFKM Key-Tabelle für einen speziellen User

Abbildung 8 skizziert eine Schlüssel-Tabelle und hebt darin jene TEK-Schlüssel hervor, welche ein User mit der ID 1001 kennen würde.
Soll ein neues Mitglied in die Gruppe hinzugefügt werden, so werden ihm vom Schlüsselserver über einen sicheren Unicast Kanal die durch seine ID bestimmten TEK's und der Gruppenschlüssel mitgeteilt.
Verlässt hingegen ein Mitglied die Gruppe, so werden alle Schlüssel, welche dieses Mitglied kannte, durch den Schlüsselserver erneuert. Daraufhin versendet der Schlüsselserver eine Multicast Nachricht, welche zwei Teile enthält. Der erste Teil besteht aus dem neuen Gruppenschlüssel, welcher W-mal in verschlüsselter Form, jeweils verschlüsselt mit einem TEK, welchen das ehemalige Mitglied nicht kannte. Da sich die ID eines jeden User an mindestens einer Stelle von der ID jedes anderen User unterscheidet, besitzt jeder User mindestens einen TEK, mit welchen er den verschlüsselten Gruppenschlüssel entschlüsseln kann. Der zweite Teil der Multicast Nachricht enthält die neuen TEK's, wobei jeder neue TEK mit dem zugehörigen alten TEK und zusätzlich mit dem neuen Gruppenschlüssel verschlüsselt wurde. Dadurch kann ein neuer TEK genau dann entschlüsselt werden, wenn ein Nutzer den alten TEK und den neuen Gruppenschlüssel kennt.
Das beschriebene Protokoll ist nicht sicher gegen Kollisionsangriffe. Besitzen beispielsweise zwei Nutzer IDs, welche bitweise komplementär sind, so könnten diese miteinander kooperieren und dadurch immer die Kenntnis von allen Schlüsseln haben und folglich auch alle zukünftigen Schlüssel berechnen, selbst wenn beide User aus der Gruppe entfernt werden [2], [1].

4 Dezentalisierte und verteilte Protokolle

4.1 Intra-Domain Group Key Management Protokoll

Das Intra-Domain Group Key Management [13] (IGKM) Protokoll ist ein dezentralisiertes Keymanagement Protokoll, welches sich dadurch kennzeichnet, dass es in mehrere Areale aufgeteilt ist. Jedes dieser Areale wird von einem Area Key Distributor (AKD) verwaltet und kann mehrere User beinhalten. Jedes Areal stellt eine lokale Gruppe dar, welche vom AKD verwaltet wird und einen gruppeninternen Schlüssel, den Area-Group-Key (AGK) enthält, welchen jedes Mitglied der Gruppe kennt und zur Arealgruppeninternen Multicast Kommunikation genutzt wird. Darüber hinaus existiert ein Domain Key Distributor (DKD), welcher dafür zuständig ist, den globalen Gruppenschlüssel

zu erzeugen und an die AKD's zu verteilen, welche diesen wiederum an die User des jeweiligen Areals verschicken. Der DKD bildet zusammen mit den AKD's eine Multicast Gruppe, dessen interne Kommunikation mit dem All-KD-Key, welcher vom DKD erzeugt und an die AKD's verteilt wird, verschlüsselt wird. Abbildung 9 skizziert eine solche

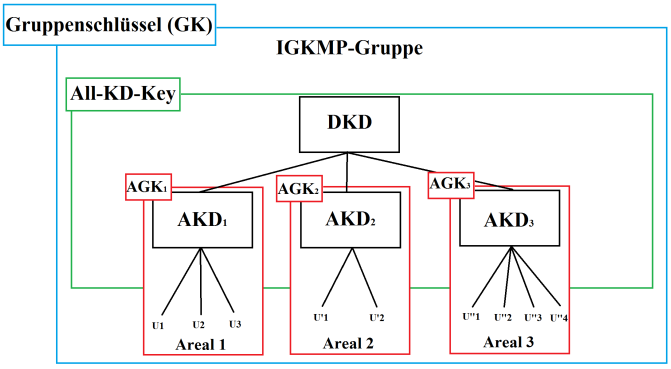

Abbildung 9. Skizze IGKMP, angelehnt an [14]

Architektur mit drei Arealen und verdeutlicht die Gültigkeitsbereiche der einzelnen Schlüssel.
Möchte ein Subjekt der Gruppe beitreten, so muss dieses einem lokalen AKD eine Anfrage schicken. Daraufhin authentifiziert das AKD dieses Subjekt und schickt die Anfrage weiter zum DKD, welches einen neuen Gruppenschlüssel erzeugt und diesen an alle AKD's, verschlüsselt mit dem All-KD-Key, versendet. Die AKD's wiederum speichern den Gruppenschlüssel und senden diesen, verschlüsselt mit dem lokalen AGK, an die lokale Gruppe. Das AKD des Areals des neuen Mitgliedes erzeugt zunächst einen neuen AGK, verschlüsselt diesen mit dem alten AGK und schickt diesen an die lokale Gruppe. Daraufhin schickt das AKD dem neuen Mitglied über eine sichere Unicast Kommunikation den neuen Gruppenschlüssel und den neuen AGK.
Verlässt ein User die Gruppe, so muss der AKD der betroffenen Region einen neuen AGK erstellen und diesen per Unicast an die restlichen Mitglieder des Areals versenden. Zusätzlich muss der AKD dem DKD den Auftrag geben einen neuen Gruppenschlüssel zu erstellen, welcher analog des Verfahrens, bei dem ein neuer Nutzer der Gruppe beitritt, durchgeführt wird [2], [14], [1].

4.2 Iolus Framework

Das Iolus Framework [15] ermölicht es ein dezentalisiertes Keymanagment aufzubauen, welches sich dadurch kennzeichnet, dass es sich hierarchisch in Untergruppen unterteilen lässt. Jede dieser Untergruppen bildet eine Multicast Gruppe mit einer eigenen Adresse und wird von einem Group security Agent (GSA) kontrolliert. Derjenige GSA, welcher die Top-Level Untergruppe kontrolliert, heißt Group security Controller (GSC) und alle anderen GSA's heißen Group security Intermediaries (GSIs). Die Gesamtheit aller Untergruppen bildet eine virtuelle Gruppe, welche keine Adresse besitzt. Außerdem sind alle GSA's in einer eigenen Multicast Gruppe. Eine beispielhafte Architektur ist in Abbildung 10 skizziert.
Möchte ein neuer User der Gruppe beitreten, so muss dieser

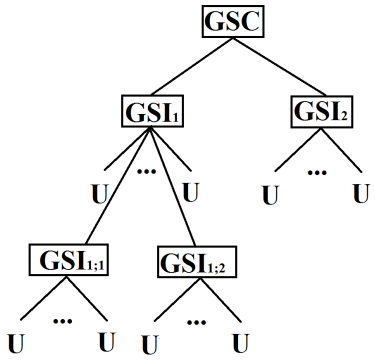

Abbildung 10. Skizze Iolus

einem GSI eine solche Anfrage senden und sich bei diesem authentifizieren. Daraufhin erzeugt das GSI einen Session Schlüssel, welcher ausschließlich für die Kommunikation mit dem neuen User verwendet wird. Im nächsten Schritt wird ein neuer Untergruppenschlüssel erstellt, welcher, verschlüsselt mit dem alten Untergruppenschlüssel, an die Untergruppe gesendet wird. Außerdem wird dem neuen User der Session Schlüssel und der neue Gruppenschlüssel über einen sicheren Unicast Kanal zugesandt.
Verlässt ein User die Gruppe, so muss der GSI, in dessen Gruppe der User lag, einen neuen Untergruppenschlüssel erzeugen und diesen für jeden User der Untergruppe mit dem zugehörigen Session Schlüssel verschlüsseln und diese verschlüsselten Nachrichten per Multicast an die Gruppe verteilen.
Möchte ein User eine Multicast Nachricht an die virtuelle Gruppe versenden, so muss dieser zunächst einen zufälligen kryptographischen Schlüsse K erzeugen und die Nachricht mit K verschlüsseln. Der Schlüssel K wird daraufhin mit dem Untergruppenschlüssel verschlüsselt und zusammen mit der verschlüsselten Nachricht per Multicast versandt. Erhält ein GSI dieser Gruppe eine solche Nachricht, so werden zwei Fälle unterschieden:
Fall 1): Der GSI ist der Controller der jeweiligen Gruppe. In diesem Fall entschlüsselt der GSI den verschlüsselten Schlüssel K und verschlüsselt diesen erneut mit dem Untergruppenschlüssel seiner Eltern-Gruppe und schickt dieser Gruppe die Nachricht mit dem neuen verschlüsselten Schlüssel K. Beispiel: Käme die Nachricht von einem User der in Abbildung 10 von GSI_1 kontrollierten Untergruppe, so würde GSI_1 den Schlüssel K mit dem Schlüssel der von GSC kontrollierten Elterngruppe verschlüsseln und zusammen mit der Nachricht an diese (also GSC und GSC_2) senden.
Fall 2): Der GSI ist ein normales Mitglied der jeweiligen Gruppe. In diesem Fall wird analog zu Fall 1 fortgefahren, wobei der GSI die Nachricht an die durch ihn kontrollierte Untergruppe sendet.
Der Vorteil dieses Protokolls ist es, dass eine Änderung der Gruppenmitgliedschaft eines Users keine Auswirkung auf die gesamte Gruppe hat, sondern lediglich auf die lokale Untergruppe [2], [1].

4.3 Distributed Logical Key Hierarchy Protokoll

Eine verteilte Variante des LKH Protokolls stellt das Distributed Logical Key Hierarchy [16] (D-LKH) Protokoll dar. Im Unterschied zum LKH Protokoll gibt es keinen zentralen Schlüsselserver, da die Schlüssel des LKH-Baumes durch die Mitglieder selber generiert werden, was darin resultiert, dass keine Entität alle Schlüssel des Baumes kennt. Im Rahmen der Schlüsselerzeugung wird der Schlüssel eines Unterbaumes durch dessen Mitglieder berechnet. Für jeden Unterbaum sei R dessen rechter Teilbaum und L dessen linker Teilbaum. Im folgenden wird für jeden dieser beiden Teilbäume ein Führer-Mitglied m_R bzw. m_L ausgewählt, woraufhin diese wie im folgenden skizziert einen Schlüssel für den jeweiligen Unterbaum aushandeln:
-m_R generiert einen Schlüssel k_{RL} und sendet diesen über einen sicheren Unicast-Kanal an den User m_L.
-m_L verschlüsselt den erhaltenen Schlüssel k_{RL} mit dem Schlüssel k_L (der Schlüssel, welcher bereits für den Teilbaum L analog des beschriebenen Verfahrens ausgehandelt wurde) und versendet diesen per Multicast an die Mitglieder des Teilbaums L. Analog verschlüsselt m_R den Schlüssel k_{RL} mit dem Schlüssel k_R und sendet diesen an den Teilbaum R. Nach Durchlauf dieser Schritte kennen alle Mitglieder des jeweiligen Unterbaumes den geheimen Schlüssel k_{RL}. Wird dieses Verfahren nach dem Bottom-up-Prinzip für alle Teilbäume des LKH-Baumes durchgeführt, so hat, nach Ablauf dieses Verfahrens, jedes Mitglied Kenntnis über den geheimen Gruppenschlüssel [14], [17].

5 Analyse der Protokolle

Tabelle 1 stellt einen Vergleich der vorgestellten zentralisierten Keymanagement Protokolle dar. Aus diesem geht hervor, dass das CFKM Protokoll im Bezug zu der Länge der Schlüsselerneuerungsnachrichten und des Speicheraufwandes für die User nur dann konkurrenzfähig mit den anderen Protokollen ist, wenn die Anzahl der Mitglieder in der Gruppe nahe an der maximal möglichen Anzahl an Mitgliedern (d.h. fast alle möglichen IDs sind in der Gruppe) ist. Vorteile bestehen in der einfachen Implementierbarkeit des Protokolls und darin, dass die Anzahl der Schlüssel, welche der Schlüsselserver speichern muss, relativ gering ist.
Das OFT Protokoll ist gegenüber dem LKH Protokoll bei dem Verlassen-Prozess um den Faktor 2 überlegen, ist jedoch nicht resistent gegenüber Kollisionsangriffen und sollte daher nur in einer Situation eingesetzt werden, in welcher den Mitgliedern einer Gruppe auch nach dessen Austritt aus dieser vertraut werden kann. Das LKH+ Protokoll ist im Bezug auf die Länge der Schlüsselerneuerungsnachricht beim Hinzufügen eines Users deutlich effizienter als das LKH Protokoll, verlangt dabei jedoch von jedem User die Berechnung von $log(n)$ Hashwerten, was beispielsweise in einem Smarthome-Szenario unerwünscht sein könnte, weshalb diese Faktoren bei der Auswahl des Protokolls berücksichtigt werden sollten [14], [2].
Ist die Gruppengröße nicht zu groß, so stellen die zentralisierten Verfahren eine gute Option dar, da diese in der Regel einfach zu implementieren und administrieren sind. Werden jedoch sehr große Gruppen mit hoher Fluktuation benötigt, so kann die Nutzung eines dezentralisierten Protokolls sinnvoll sein, da hierbei die Last auf mehrere Entitäten aufgeteilt wird. Ein großes Problem des Iolus Frameworks besteht darin, dass jede Nachricht durch die GSAs übersetzt werden muss und somit für jede, in der Multicast Gruppe gesendete, Nachricht Arbeit für jeden GSA entsteht, was dazu führt, dass in einer Gruppe, in welcher viel Kommunikation stattfindet, der GSA ein limitierender Faktor darstellen kann. Ein Vorteil liegt hingegen darin, dass bei einem Hinzufügen oder Entfernen eines Nutzers nur die jeweilige Untergruppe involviert ist und die restlichen Mitglieder der gesamten Gruppe davon unberührt bleiben, also keine Berechnungen durchführen müssen. Demgegenüber besteht das Problem der notwendigen Übersetzungen bei dem IGKMP Protokoll nicht, jedoch wird hier bei jeder Gruppenmitgliedschaftsänderung jeder User involviert. Darüber hinaus sind die beiden Protokolle schwieriger zu implementieren und administrieren, als bei der zentralisierten Variante, jedoch haben dezentralisierte Protokolle oftmals den Vorteil, dass diese robuster gegenüber von Fehlern einzelner Entitäten sind (single point of failure). Verteilte Keymanagement Protokolle, wie zum Beispiel das D-LKH Protokoll, weisen ebenfalls diese Probleme auf und bieten darüber hinaus eine schlechte Skalierbarkeit und ein niedrigeres Sicherheitslevel, da jeder User potentiell Schlüssel für den Schlüsselbaum der Gruppe erstellen könnte, dieser User aber nicht notwendiger Weise die Sicherheitsmaßstäbe eines Schlüsselservers aufweist. Erzeugt ein solches Mitglied beispielsweise Zufallszahlen mit niedriger Entropie, so wären alle Nachrichten, welche mit einem von diesen Mitglied generierten Schlüssel verschlüsselt wurden, angreifbar. Ein mögliches Szenario, in welchem ein solchen Protokoll dennoch sinnvoll ist, stellt beispielsweise ein kabelloses Netzwerk dar, in welchen nicht jeder User mit jedem anderen User direkt kommunizieren kann. Ist es allerdings möglich ein (de)zentralisiertes Protokoll ohne Einschränkungen zu verwenden, so sollte ein solches im allgemeinen bevorzugt werden [14], [2], [18], [7], [19].

6 Zusätzliche Anmerkungen

6.1 Sicherheitsrelevante Anmerkungen

Angriffe, beispielsweise der man-in-the-middle Angriff, oder ein Replay-Angriff, sind Angriffe, welche im Bereich der Unicast Kommunikation betrachtet werden müssen, sollten dies oftmals ebenfalls im Bereich der Multicast Kommunikation.
Ein man-in-the-middle Angriff könnte im Bereich des Multicast Keymanagements beispielsweise dann durchgeführt werden, wenn eine Entität der Gruppe mit einem neuen User eine Unicast Verbindung aufnimmt, um diesen User im weiteren Verlauf die relevanten Informationen zukommen zulassen. Gelänge es einem Angreifer einen solchen man-in-the-middle Angriff erfolgreich durchzuführen, so bekäme dieser den Gruppenschlüssel, ohne das dies erkannt werden würde. Um einen solchen Fall auszuschließen ist es ratsam einen authentifizierten Schlüsselaustausch durchzuführen, sodass sichergestellt ist, dass zwischen dem User und der Entität der Gruppe kein Dritter liegt, welcher die Kommunikation zwischen den Beiden verändert [20].
Ein Replay-Angriff kennzeichnet sich dadurch, dass ein Angreifer eine kryptographisch gesicherte Nachricht abhört und diese zu einem späteren Zeitpunk selber versendet und dadurch eine für den Empfänger valide Nachricht

Tabelle 1
Vergleich-zentralisierte Keymanagement-Protokolle [14], [2]
n: Anzahl Mitglieder in der Gruppe; I: Anzahl Bits in ID

	LKH	LKH+	OFT	CFKM
Hinzufügen (Multicast)	$log_2(n) - 1$	1	$log_2(n) + 1$	2*I
Hinzufügen (Unicast)	$log_2(n) + 1$	$log_2(n) + 1$	$log_2(n) + 1$	I+1
Verlassen (Multicast)	$2 * log_2(n)$	$2 * log_2(n)$	$log_2(n) + 1$	2*I
Speicheraufwand (Server)	n+2	n+2	2*n-1	2*I+1
Speicheraufwand (User)	$log_2(n) + 1$	$log_2(n) + 1$	$log_2(n) + 1$	I+1
Kollisionsresistent	ja	ja	nein	nein

verschickt hat, ohne den dahinter liegenden kryptographischen Schlüssel zu kennen. Beispielsweise könnte im Iolus-Framework ein Angreifer, welcher zeitweise in der Gruppe ist, eine Multicast Schlüssel-Erneuerungsnachricht des GSI's seiner zugehörigen Untergruppe speichern und diese zu einem späteren Zeitpunkt, zu welchem der Angreifer nicht mehr in der Gruppe ist, per Multicast an die Untergruppe versendet. Ist kein Schutz gegen einen solchen Replay-Angriff gegeben, so würden die Mitglieder der Untergruppe daraufhin einen alten, von dem Angreifer bekannten, Gruppenschlüssel zur weiteren Kommunikation verwenden. Eine Gegenmaßnahme gegen einen solchen Replay-Angriff wäre beispielsweise die Nutzung von Sequenznummern, welche in die kryptographische Nachricht verbaut werden, damit diese Nachricht zu einem späteren Zeitpunkt keine Gültigkeit mehr besitzt. Da im Bereich der Multicast-Verschlüsselung mehrere Mitglieder der Gruppe gleichzeitig Nachrichten schicken können, ist es notwendig diese Sequenzunummern zu synchronisieren [20], [21].
Einen besondere Schwachstelle des Keymanagements stellen die Gruppenmitglieder einer Multicast Gruppe dar, da jedes dieser Mitglieder in der Lage ist, die verschlüsselten Nachrichten der Gruppe zu entschlüsseln. Daraus folgt, dass jedes Mitglied einen potentiellen Angriffsvektor für einen Angreifer darstellt, weshalb die Sicherheit der Gruppe davon abhängt, dass jedes Gruppenmitglied sicher gegen mögliche Angriffe ist, da ein Angreifer in der Regel den einfachst möglichen Weg für einen Angriff sucht. Mögliche Angriffe gegen die einzelnen Gruppenmitglieder hängen von der Beschaffenheit eines solchen Mitgliedes ab und sind daher anwendungsspezifisch. Ist das Mitglied beispielsweise eine Chipkarte, so könnte der geheime Gruppenschlüssel beispielsweise durch Seitenkanalangriffe erhalten werden. In dem Multicast-Szenario einer Video-Konferenz ist hingegen der menschliche-Benutzer selber ein potentielles Risiko, da ein Angreifer beispielsweise durch Social Engineering Maleware auf das Gerät des Benutzers laden könnten und dadurch Zugang zu der geheimen Gruppenkommunikation hätten. Ein weiteres Problem hierbei liegt darin, dass ein solcher Angriff sehr schwer zu registrieren ist, da hierfür jedes Mitglied untersucht werden müsste. Schutz gegen solche Angriffe kann nur dadurch erreicht werden, indem die einzelnen Mitglieder geschützt werden, bzw. nur solche, welche einen angemessenen Schutz aufweisen, in die Gruppe gelassen werden. Im Beispiel der Chipkarten könnte beispielsweise eine Zertifizierung (z.B. nach der Common Criteria) gefordert werden [7].
Einen weiteren Angriff auf die Keymanagement Protokolle stellen Denial of Service Angriffe dar, bei welchen die Verfügbarkeit einer oder mehrerer Instanzen beeinträchtigt wird. Wird beispielsweise bei der Verwendung eines zentralisierten Keymanagement-Protokolls der Schlüsselserver durch eine Denial of Service Attacke angegriffen und ist nicht mehr erreichbar, so können keine Änderungen der Gruppenstruktur (Mitglieder hinzufügen / entfernen) mehr durchgeführt werden. Soll ein Mitglied aus der Gruppe entfernt werden, so ist dies nicht möglich, da der Schlüsselserver nicht in der Lage ist, Schlüsselerneuerungs-Nachrichten zu versenden. Eine Abschwächung dieses Problems kann beispielsweise durch die Nutzung eines dezentralisierten Keymanagement Protokolls erreicht werden.

6.2 User verlässt die Gruppe

Bisher wurde unter dem Verlassen der Gruppe verstanden, dass ein User explizit das Verlassen der Gruppe initialisiert, oder durch eine autorisierte Instanz aus der Gruppe entfernt wird. Dies würde allerdings das Problem eröffnen, dass ein Gruppenmitglied, welches die Gruppe implizit verlässt (beispielsweise weil es kaputt gegangen ist), ohne eine „Verlassen"-Operation zu initialisieren, weiterhin in der Gruppe erfasst werden würde, wodurch die Multicast Gruppe größer wäre als nötig und dadurch bei den Gruppenoperationen ein erhöhter Aufwand entstünde. Eine Möglichkeit, um solche User zu detektieren wäre es, in regelmäßigen Abständen eine „Mitglieder aktualisieren"-Nachricht an die Gruppe zu senden, wobei jedes Mitglied der Gruppe, welches in der Gruppe bleiben möchte, auf diese Nachricht antworten müsste. Bleibt diese Antwort für eine definierte Zeit aus, so würde ein solcher User aus der Gruppe entfernt werden [22].

6.3 Inaktiver User

Ist ein User der Gruppe nicht aktiv, beispielsweise um Strom zu sparen, so werden die folgenden Fälle unterschieden:

Der User ist nicht aktiv, speichert die eingehende Kommunikation jedoch ab: In diesem Fall muss der User, sobald er wieder aktiv ist, die Schlüsselerneuerungsnachrichten in chronologischer Reihenfolge abarbeiten, um den aktuellen Gruppenschlüssel zu berechnen. Abhängig vom verwendeten Protokoll kann auch die Bearbeitung einer Teilmenge dieser Nachrichten ausreichend sein. Im Iolus Framework ist es beispielsweise ausreichend bei jener Schlüsselerneuerungsnachricht zu beginnen, welche bei der letzten „Verlassen"-Operation gesendet wurde, da diese Nachricht den „neuen" Untergruppenschlüssel verschlüsselt mit dem vom User bekannten Session Schlüssel enthält.

Der User ist nicht aktiv und speichert keine eingehende Kommunikation ab: Damit der User wieder an den aktuellen Gruppenschlüssel gelangen und weiterhin in der Gruppe teilnehmen kann, sind beispielsweise die folgenden Möglichkeiten denkbar:
-Der User verlässt die Gruppe und tritt wieder ein. Dadurch hätte der User den aktuellen Gruppenschlüssel. Allerdings sind diese Operationen relativ kostspielig und erfordern je nach verwendeten Multicast Keymanagement Protokoll Berechnungen von einem Teil, oder gar allen Mitgliedern der Gruppe.
-Der User stellt eine Anfrage auf das Zusenden der für ihn relevanten Informationen. Dies ist allerdings nur dann möglich, wenn es im Bereich des Protokolls eine Instanz gibt, welche alle für den User relevanten Informationen kennt. Beispielsweise könnte der Schlüsselserver im Rahmen des LKH Protokolls eine solche Anfrage beantworten, indem dieser den User per gesicherter Unicast Kommunikation jene Schlüssel zuschicken würde, welche in dem Schlüsselbaum auf dem Weg vom User zur Wurzel liegen [23].

7 Beispiel-Szenario

Ein mögliches Szenario für das Multicast Keymanagement im Kontext des Internet der Dinge kann in der Kommunikation zwischen mehreren Lichtschaltern und mehreren Lampen gesehen werden. Dabei wird davon ausgegangen, dass jedes dieser Entitäten einen User darstellt und alle diese User in einer Multicast Gruppe (siehe Kapitel 2.1) liegen. Gefordert sei, dass die einzelnen User untereinander mittels Multicast verschlüsselt und authentifiziert kommunizieren können. Eine Möglichkeit zur Realisierung dieser Anforderung besteht in der Nutzung des Multicast Keymanagements, durch welches jedes Gruppenmitglied einen geheimen Schlüssel (den Gruppenschlüssel) erhält, mit Hilfe dessen es jeden User möglich ist, verschlüsselte Nachrichten in die Multicast Gruppe zu schicken, wobei die Tatsache, dass ein User den geheimen Schlüssel kennt, beweist, dass dieses ein authentifizierter Nutzer ist.
Ein für diesen Fall geeignetes Multicast Keymanagement Protokoll wäre beispielsweise das in Kapitel 3.1 beschriebene LKH-Protokoll, da dieses vergleichsweise einfach zu realisieren und administrieren ist, die Anzahl der User relativ gering ist und ein Denial of Service Angriff auf den Schlüsselserver unwahrscheinlich scheint. Damit dieses verwendet werden kann, wird zusätzlich ein Schlüsselserver benötigt, welcher in der gleichen Multicast Gruppe liegt. Zusätzlich wird angenommen, dass jeder User eine eindeutige ID und ein geheimes Passwort besitzt und in der Lage ist modulare Exponentiation effizient zu berechnen.
Soll nun ein User, also ein Lichtschalter, oder eine Lampe, in die gesicherte Multicast Gruppe hinzugefügt werden, so ist es notwendig, dass der Schlüsselserver eine gesicherte Kommunikation zu dem User aufnehmen kann. Dies kann dadurch realisiert werden, dass eine authentifizierte Person das Passwort und die ID des Users in dem Schlüsselserver manuell einträgt. Daraufhin könnte der User eine Beitrittsanfrage an den Schlüsselserver senden, wobei daraufhin ein authentifizierter Schlüsselaustausch, analog zu der Beschreibung in Kapitel 2.3, durchgeführt werden könnte. Ist dieser Austausch erfolgreich, so besitzen beide Kommunikationspartner einen geheimen Schlüssel zur Unicast Kommunikation und zusätzlich ist der jeweils andere Kommunikationspartner authentifiziert worden. Aus Sicht des Schlüsselservers gilt der User als authentifiziert, da der Schlüsselserver einen Eintrag für die ID und das Passwort des Users hat und der User durch den erfolgreichen Schlüsselaustausch bewiesen hat, dass dieser das Passwort kennt. Andersherum gilt der Schlüsselserver für den User durch den erfolgreichen Schlüsselaustausch als authentifiziert, da dieser bewiesen hat, dass er das geheime Passwort kennt.
Abschließend kann der neue User nach dem in Kapitel 2.3 beschriebenen Verfahren auf Grundlage des sicheren Unicast-Kanals in die Gruppe integriert werden.

8 Fazit

Die Verwendung von Multicast-Verschlüsselung und damit einhergehend dem Multicast Keymanagment kann in vielen Situationen sehr sinnvoll sein. Ein Multicast Keymanagement Protokoll sollte individuell nach dem individuellen Gegebenheiten ausgewählt werden. So ist in einer relativ kleinen bis mittelgroßen Gruppe ein zentralisiertes Protokoll aufgrund seiner Einfachheit zu bevorzugen, wohingegen in einer großen Gruppe ein dezentralisiertes Protokoll verwendet werden sollte. Auch das zu erreichende Sicherheitslevel und die Beschaffenheit der einzelnen Gruppenmitglieder(z.B. Rechenstärke) sind wichtige zu berücksichtigende Faktoren. Verteilte Schlüsselmanagementprotokolle haben einige Nachteile, können jedoch in speziellen Situationen notwendig sein, sofern andere Protokolle nicht angewandt werden können.
Von großer Bedeutung bei der Betrachtung der Sicherheit von Multicast Protokollen sind bereits aus dem Unicast-Fall bekannte Angriffe wie zum Beispiel Man-in-the-middle Angriffe, oder Replay-Angriffe, gegen welche geeignete Gegenmaßnahmen getroffen werden sollten. Einen sehr schwer zu kontrollierenden und für einen Angreifer oftmals vergleichsweise attraktiven Angriffsvektor stellen die Gruppenmitglieder selber dar, da die Kompromittierung eines einzelnen Gruppenmitgliedes mit der Kompromittierung der Gruppe einhergeht. Abschließend bleibt zu sagen, dass die größten Gefahren im Bereich der Multicast-Verschlüsselung weniger in den Keymanagement Protokollen selber liegen, sondern vielmehr in den restlichen Rahmenbedingungen, wie zum Beispiel dem Aufbau einer sicheren Unicast Verbindung und den einzelnen Gruppenmitgliedern.

Literatur

[1] X. H. Bibo Jiang, "A survey of group key management," *International Conference on Computer Science and Software Engineering*, 2008.

[2] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 309–329, Sep. 2003. [Online]. Available: http://doi.acm.org/10.1145/937503.937506

[3] S. Q. Li and Y. Wu, "A survey on key management for multicast," in *Information Technology and Computer Science (ITCS), 2010 Second International Conference on*, July 2010, pp. 309–312.

[4] C. Duma, *Security and Efficiency Tradeoffs in Multicast Group Key Management*, ser. Linköping studies in science and technology: Thesis. Univ., 2003. [Online]. Available: https://books.google.de/books?id=504sPQAACAAJ

[5] M. Singhal, R. Bai, Y. Lin, Y. Wang, M. Yang, and Q. Zhang, "Key management protocols for wireless networks."

[6] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY*, 1992, pp. 72–84.

[7] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," United States, 1999.

[8] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 68–79, Oct. 1998. [Online]. Available: http://doi.acm.org/10.1145/285243.285260

[9] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The versakey framework : versatile group key management," *IEEE Journal on selected areas in communications*, vol. 17, no. 9, pp. 1–16, 1999.

[10] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Trans. Softw. Eng.*, vol. 29, no. 5, pp. 444–458, May 2003. [Online]. Available: http://dx.doi.org/10.1109/TSE.2003.1199073

[11] M. Y. Malik, "Efficient group key management schemes for multicast dynamic communication systems," Cryptology ePrint Archive, Report 2012/628, 2012, http://eprint.iacr.org/2012/628.

[12] G. HORNG, "Cryptanalysis of a key management scheme for secure multicast communications," *IEICE transactions on communications*, vol. 85, no. 5, pp. 1050–1051, may 2002. [Online]. Available: http://ci.nii.ac.jp/naid/110003219431/en/

[13] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure group communications for wireless networks," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, 2001, pp. 113–117 vol.1.

[14] Y. Challal, A. Bouabdallah, and H. Seba, "A taxonomy of group key management protocols: Issues and solutions," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 1, no. 6, pp. 170 – 182, 2007. [Online]. Available: http://iastem.com/Publications?p=6

[15] S. Mittra, "Iolus: A framework for scalable secure multicasting," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 277–288, Oct. 1997. [Online]. Available: http://doi.acm.org/10.1145/263109.263179

[16] O. Rodeh, K. Birman, and D. Dolev, "Optimized group rekey for group communications systems," Ithaca, NY, USA, Tech. Rep., 1999.

[17] A. Mehdizadeh, F. Hashim, and M. Othman, "Lightweight decentralized multicast–unicast key management method in wireless {IPv6} networks," *Journal of Network and Computer Applications*, vol. 42, pp. 59 – 69, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804514000812

[18] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 11-12, pp. 2314–2341, Sep. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2007.04.009

[19] F. Du, L. M. Ni, and A. H. Esfahanian, "Towards solving multicast key management problem," in *Computer Communications and Networks, 1999. Proceedings. Eight International Conference on*, 1999, pp. 232–236.

[20] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm, "Multicast security (msec) group key management architecture," Internet Requests for Comments, RFC Editor, RFC 4046, April 2005.

[21] G. Steel and A. Bundy, "Attacking group multicast key management protocols using coral," *Electronic Notes in Theoretical Computer Science*, vol. 125, no. 1, pp. 125 – 144, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066105000782

[22] B. Wu, J. Wu, and Y. Dong, "An efficient group key management scheme for mobile ad hoc networks," *Int. J. Secur. Netw.*, vol. 4, no. 1/2, pp. 125–134, Feb. 2009. [Online]. Available: http://dx.doi.org/10.1504/IJSN.2009.023431

[23] B. C. T. Hardjono and I. Monga, "Intra-domain group key management for multicast security," Intra-domain Group, IETF Internet draf, September 2000.

[24] Y. Baddi and M. D. E.-C. E. Kettani, "Key management for secure multicast communication: A survey," in *Security Days (JNS3), 2013 National*, April 2013, pp. 1–6.

[25] R. C. Gangwar, "Secure and efficient decentralized group key establishment protocol for robust group communication."

[26] B. Briscoe, *MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 301–320.

[27] M. A. S. Jr., P. S. Barreto, C. B. Margi, and T. C. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, vol. 54, no. 15, pp. 2591 – 2612, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861000112X

[28] L. R. Dondeti, S. Mukherjee, and A. Samal, "A distributed group key management scheme for secure many-to-many communication," 1999.