Universität Karlsruhe (TH)
Institut für Telematik

TeleMatics Technical Reports

Netzsicherheit und Hackerabwehr
Seminar WS08/09

Denise Dudek, Stephan Krause, Lars Völker, Christoph Werle
Prof. Dr. Martina Zitterbart
{dudek,krause,lars.voelker,werle,zit}@tm.uka.de

25. März 2009

TM-2009-2

ISSN 1613-849X
http://doc.tm.uka.de/tr/

Institute of Telematics, University of Karlsruhe
Zirkel 2, D-76128 Karlsruhe, Germany

Vorwort

Wie bereits in vergangenen Jahren, wurden auch in diesem Semester im Rahmen des Seminars *Netzsicherheit und Hackerabwehr* aktuelle sicherheitsrelevante Fragestellungen von Studenten bearbeitet. Das breite Spektrum von Themen des Seminars umfasste dabei beispielsweise den Einsatz von Sicherheitsmechanismen und -protokollen in ressourcenbeschränkten Sensornetzen – einem Thema, an dem das Institut für Telematik im Auftrag des BSI in dem Projekt FleGSens auch aktuell forscht. Weitere Arbeiten befassten sich mit der Sicherheit der Kommunikationsinfrastruktur des heutigen Internets wie z.B. dem Routing und der Namensauflösung. Letztere geriet insbesondere durch einen von Dan Kaminsky entwickelten, sehr effizienten Angriff auf das Domain Name System in die Schlagzeilen. Neben weiteren Themen wurden auf Grund der anhaltenden Popularität von Onlinespielwelten, wie z.B. World of Warcraft, dort auftretende Sicherheitsprobleme und mögliche Gegenmaßnahmen betrachtet.

Der vorliegende Seminarband fasst die Ausarbeitungen der Studenten in Form eines technischen Berichts zusammen.

Inhaltsverzeichnis

TESLA - Sicheres Broadcasting in Sensornetzen

Tobias Zepf

Sensornetze spielen aufgrund ihrer vielfältigen Anwendungsmöglichkeiten zum Beispiel zur Überwachung eine immer wichtigere Rolle in der heutigen Welt und werden somit auch als Ziel von Angriffen attraktiver. Da Broadcasting vor allem in drahtlosen Sensornetzen zur Kommunikation benutzt wird [Gola02] sollte gewährleistet sein, dass die Datenübertragung zwischen dem Sender und den Empfängern sicher und zugleich auch effizient ist. Das Protokoll „TESLA“ (Timed Efficient Stream Loss-tolerant Authentication) und die für Sensornetze besser geeignete Variante μTESLA leisten das Gewünschte. Beide Protokolle werden in dieser Arbeit vorgestellt.

1 Sensornetze

Als Sensornetz bezeichnet man ein Netz von mehreren Sensorknoten, die räumlich verteilt sind. Diese Knoten sind Kleinstrechner und nur wenige Zentimeter oder teilweise auch Millimeter groß. Die Kommunikation kann drahtlos oder drahtgebunden erfolgen, wobei in den

Abbildung 1: Größe eines Smart Dust Sensorknotens zur Veranschaulichung [Gola02]

meisten Anwendungen die drahtlose Kommunikation gewählt wird, da sie deutlich einfacher zu realisieren ist. Die Aufgabe von solchen Netzen ist zum Beispiel die Überwachung von realen, physischen Bedingungen wie Temperatur, Bewegung etc. Die geringe Größe erlaubt eine kostengünstige Herstellung und den Einsatz großer Mengen von Sensorknoten. [Bagc07].

Sensornetze haben ihren Ursprung im militärischen Bereich. Schon im Jahr 1949 begann die amerikanische Navy mit den Plänen, feindliche U-Boote frühzeitig mit Hilfe von Sonar zu erkennen. In den 50er und 60er Jahren wurden rund um den Amerikanischen Kontinent Sensorknoten „versenkt“ [Glob]. Auch heute wird im Militär noch an Sensornetzen geforscht und gearbeitet. Dort werden Sensornetze in der ABC-Kampfstofferkennung eingesetzt. Im zivilen Bereich ergeben sich interessante Anwendungsszenarien. Ein Beispiel die Medizin. [Krae06] Hier können Sensornetze benutzt werden um die Vitalfunktionen zu überprüfen. Ein weiteres Anwendungsgebiet ist die intelligente Verkehrsüberwachung, bei der die Sensorknoten in Autos drahtlos miteinander kommunizieren, um vor Staus, Unfällen, Glätte und anderen Einflüssen zu warnen.

Derzeit forscht man am Fraunhofer Institut für Zuverlässigkeit und Mikrointegration (IZM) am Konzept des 'Smart Dust'. Der intelligente Staub soll unter anderem Temperaturen an

schwer zugänglichen Orten oder den Zustand von Pflanzen auf Feldern überwachen. Dabei sollen die Sensorknoten so klein werden, dass Sie sich wie Staub beliebig ausbreiten lassen. Derzeit hat man die Knoten auf 6 mm Kantenlänge miniaturisiert, wie in Abbildung 1 zu sehen ist. Das IZM rechnet damit, die ersten Smart Dust Sensornetze in zwei bis drei Jahren marktreif und einsatzfähig zu haben [BäMü08].

1.1 Kommunikation zwischen Sensorknoten

Point-to-point Kommunikation ist in Sensornetzen nicht geeignet, da ein Sensorknoten meist alle anderen Knoten in der Umgebung gleichzeitig benachrichtigen will und nicht nur einen einzigen. Zudem nutzen die meisten Sensornetze drahtlose Netze zur Kommunikation, da deren Realisierung deutlich einfacher und in den meisten Anwendungsszenarien unumgänglich ist. Dabei werden hauptsächlich die Standards 802.15.4 [Socic], 802.11 [Socia] und Bluetooth [Socib] genutzt [Wels05]. 802.11 ist der populärste Standard und hat im Idealfall eine Reichweite von 100 - 300m im Freien [Kompa] und hohe Übertragungsraten [Kompb]. Dagegen ist „Bluetooth [...] mit geringen Hardwarekosten, niedrigem Stromverbrauch und Echtzeitfähigkeit in den Bereichen Sprachübertragung, Audio-Video-Lösungen und Adhoc-Verbindungen zwischen Kleinstgeräten besser geeignet“ [Kompa] und bietet Datenraten von 720 Kbit/s [Wels05]. 802.15.4, auch 'Zigbee' genannt, wurde speziell für drahtlose Netze mit niedrigen Datenraten entwickelt. Sie beträgt 250 Kbit/s bei geringen Latenzzeiten und Strombedarf [IIS]. Daher ist Zigbee besonders für Sensornetze geeignet, da hier Strom nur sehr begrenz zur Verfügung steht. Alle drei Standards haben aber eines gemeinsam. Die Teilnehmer kommunizieren häufig über Broadcasts miteinander, um unter anderem Softwareupdates zu verteilen, Netzwerkabfragen zu verschicken oder Befehle zu verbreiten [18t07]. Broadcasting birgt aber einige nicht zu unterschätzende Gefahren, auf die im nächsten Abschnitt eingegangen wird.

2 Sicherheit beim Broadcasting

Das Ziel von Sicherheitsarchitekturen ist es, potentiellen Angreifern entgegenzuwirken. Dabei ist ein Angreifer eine Entität, die mit ihren Aktionen so viel Schaden wie möglich anrichten will. Daher muss vor allem in denjenigen Umgebungen, die Broadcasts häufig benutzen, auf Sicherheit geachtet werden, da hier sonst eine Vielzahl von Opfern mit einem Paket erreicht werden kann. Die hauptsächlichen Herausforderungen beim Broadcasting sind:

- **Zuverlässigkeit bei der Datenübertragung**

  Geht ein Paket beim Broadcasting verloren, können die Sender nicht wie bei Point-to-point Übertragungen eine erneute Übertragung anfordern. Der Sender würde mit Anfragen zur erneuten Übertragung überhäuft, was zur Überlastung führen kann.

- **Authentifizierung**

  Mithilfe von authentifizierten Nachrichten garantiert werden, dass eine Nachricht auch tatsächlich vom angegebenen Sender stammt [BeNS05]. Beim Broadcasten kann unter Umständen ein Paket für mehrere Millionen Empfänger bestimmt sein. Das birgt natürlich auch ein großes Risiko. Ein Sender kann in der Lage dazu sein, alle Empfänger mit schädlichem Code zu erreichen. Ein Beispiel hierfür ist die Windows Teardrop attack [Micr06], die Windows NT Systeme abstürzen lies. Dieser Attacke kann man mit Authentifizierung vorbeugen.

- **Datenintegrität**

  Datenintegrität bedeutet, dass Daten nicht von Dritten unberechtigt modifiziert oder gelöscht worden sind. Normalerweise ist es aber beim Broadcasten nicht vorgesehen, verlorene oder gelöschte Pakete erneut zu übertragen, da von vorneherein darauf geachtet

werden muss, dass es eine Robustheit gegen Paketverlust gibt. Daher reicht Authentifizierung der Daten aus, um die Integrität zu gewährleisten. [PeTy03]

- **Data freshness**

  Messergebnisse, die in Sensornetzen gesendet werden, sollen zeitnah ankommen. Daher reicht Authentizität nicht aus, es muss zusätzlich die Aktualität der Daten (Data freshness) garantiert werden, um zu verhindern, dass zum Beispiel ein Feind alte abgefangene Daten verschickt, um dem Empfänger zu verwirren.

2.1 Kryptographische Grundlagen

Um Authentizität beim Broadcasting zu erreichen sind kryptographische Verfahren nötig. Die Grundlagen dazu werden im folgenden detailliert vorgestellt.

2.1.1 Symmetrische und Asymmetrische Kryptographie

Bei Symmetrischer Kryptographie, auch Ein-Schlüssel-Kryptosystem genannt, benötigen Sender und Empfänger den gleichen, vor Dritten geheim zu haltenden Schlüssel. Jeder der N Teilnehmer in einem Netzwerk muss daher $N - 1$ Schlüssel speichern. Sender und Empfänger müssen „sich gegenseitig vertrauen, dass keiner den gemeinsamen Schlüssel preisgibt“ [BeNS05].

Im Gegensatz dazu muss bei asymmetrischer Kryptographie, auch Public-key-Kryptographie genannt, nur ein privater und ein öffentlicher Schlüssel vorgehalten werden. Anschaulich funktionieren Public-key-Verfahren wie folgt: Wenn Alice eine Nachricht an Bob schicken möchte, benötigt Alice zuerst Bobs öffentlichen Schlüssel. Dieser kann auf einem Server liegen oder direkt von Bob angefordert werden. Alice wendet den öffentlichen Schlüssel auf die Nachricht an und erhält einen Geheimtext. Da nur Bob den zu seinem öffentlichen Schlüssel passenden Privatschlüssel kennt, kann auch nur er die Nachricht entschlüsseln. Dabei ist darauf zu achten, dass aus dem öffentlichen Schlüssel von Bob nicht auf den privaten geschlossen werden kann, was die Schlüsselerzeugung komplexer als bei symmetrischen Verfahren macht, bei denen einfache Pseudo-Zufallszahlen ausreichend sind[1].

Der Vorteil gegenüber symmetrischen Varianten ist, dass spontan vertraulich kommuniziert werden kann und nicht erst ein vertraulicher Schlüssel vereinbart werden muss. Somit muss jeder Teilnehmer auch nur seinen öffentlichen und privaten Schlüssel speichern. Allerdings benötigt man bei asymmetrischer Kryptographie eine Vertrauensinfrastruktur, um die Authentizität der Zuordnung von Schlüsseln zur Identität zu garantieren, da ansonsten ein Angreifer seinen eigenen öffentlichen Schlüssel als den eines anderen Teilnehmers ausgeben kann. Dies würde ihm ermöglichen die Nachrichten an den Teilnehmer, die mit Hilfe des öffentlichen Schlüssels des Angreifers geschützt sind, mitzulesen.[BeNS05].

2.1.2 Message authentication code

Bei point-to-point Datenübertragungen ist eine Authentifizierung einfacher zu realisieren als bei Broadcasts. Sender und Empfänger teilen sich dabei einen geheimen Schlüssel um einen „message authentication code“ (MAC) zu berechnen. Als MAC bezeichnet man „eine Familie

$$\{h_K \mid K \in \mathcal{K}\}$$

[1] Pseudo-Zufallszahlen sollen so aussehen, als ob sie zufällig vom Computer gewählt worden und möglichst gleichverteilt sind. Sie werden von Pseudo-Zufalls-Funktionen berechnet [BeNS05]

von Hashfunktionen wobei $\mathcal{K}$ eine Menge von Schlüsseln ist“ [Wätj08]. Dabei ist für beliebige Schlüssel $K \in \mathcal{K}$ und beliebige Eingaben variabler Länge x der Wert $h_k(x)$ leicht zu Berechnen und hat eine feste Bitlänge. Das Ergebnis wird MAC genannt. Der Sender berechnet also einen MAC und hängt diesen an die Nachricht. Nach Erhalt berechnet der Empfänger diesen erneut und vergleicht das Ergebnis mit dem Wert des Senders. Stimmt dieser überein, ist der Empfänger sicher, dass die Nachricht vom Sender stammt. „Ein MAC garantiert also die Authentizität einer Nachricht, nicht aber ihre Geheimhaltung“, [Wätj08].

Allerdings ist symmetrische MAC Authentifizierung beim Broadcasting nicht sicher, da jeder Empfänger denselben Schlüssel kennt und somit in die Rolle eines Senders schlüpfen kann. Das impliziert, dass für sicheres Broadcasting ein asymmetrischer Mechanismus benötigt wird. Digitale Signaturen sind zum Beispiel solch ein Mechanismus. Allerdings ist die Erzeugung von solchen Signaturen zu aufwändig und teuer für die Sensorknoten. Durch die geringen Rechenkapazitäten eines Knotens werden diese zu stark mit der komplexen Berechnung ausgelastet. Untersuchungen zufolge ist asymmetrische Verschlüsselung bis zu 22.000 mal teurer als symmetrische [Blaß07]. Das heißt asymmetrische Verfahren sind einsetzbar, verbrauchen aber viel mehr Energie und Rechenzeit als symmetrische Verfahren. Die symmetrische Verschlüsselung kostet sogar deutlich weniger Energie als der Versand einer Nachricht selbst. Zudem werden hinreichend lange Schlüssel benötigt, was dazu führt, dass für Signaturen zuviel Platz benötigt wird. Nutzt man zum Beispiel dass RSA-Verfahren, werden momentan 640 Bit Schlüssel empfohlen [Blaß07]. In Zukunft werden diese aber auch nicht ausreichen und es muss auf noch längere Schlüssel zurückgegriffen werden.

In Sensornetzen sind aber gerade Energie und Rechenkapazität nur begrenzt vorhanden, wodurch Verfahren wie RSA aufgrund ihres Overheads nicht in Frage kommen. Aus diesen Gründen wurde das Protokoll TESLA entwickelt, welches im folgenden Abschnitt genauer betrachtet wird.

3 TESLA

TESLA basiert rein auf symmetrischer Verschlüsselung, bietet aber mithilfe der Zeit auch die asymmetrische Eigenschaft, die für Broadcasting unabdingbar ist [PeTy03].

3.1 Anforderungen

Die Anforderungen an Broadcast Authentifizierung in Sensornetzen sind:

- Geringe Berechnungszeit für die Generierung und Verifizierung von Informationen
- Geringer Speicherbedarf für Schlüssel
- Robustheit gegen den Verlust von Paketen
- Skalierbarkeit auf eine sehr große Anzahl von Empfängern

TESLA benötigt außerdem noch folgende spezifische Eigenschaften:

- Die Uhrzeiten von Sender und Empfänger müssen ungefähr synchronisiert sein
- Sender oder Empfänger müssen Nachrichten zwischenspeichern können

3.2 Kryptographische Grundlagen für TESLA

Für das Verständnis der Funktionsweise von TESLA sind noch weitere kryptographische Begriffe zu klären.

3.2.1 Commitment - Funktion

Verbindlichkeit (Commitment) bedeutet, dass ein Empfänger nach dem Erhalt einer Nachricht Dritten nachweisen kann, dass die Nachricht tatsächlich vom Sender stammt. Dazu wird eine Einweg-Funktion benötigt, die recht leicht zu berechnen, aber mit heutigen Computern nicht in einem vernünftigen Zeitaufwand zu invertieren ist. Diese sollte auch stark Kollisionsresistent sein. Das heißt für zwei unterschiedliche Werte gibt es keinen gleichen Funktionswert. Mit solch einer Funktion, im folgenden F genannt, wird nun zu einem geheimen Wert s der Schlüssel $F(s) = c$ berechnet und dieses c anschließend veröffentlicht. Zu diesem Zeitpunkt ist es für Dritte aufgrund der Eigenschaften von F nicht möglich, s herauszufinden . Erst mit dem veröffentlichen von s, auch „Öffnen der Verbindlichkeit“ genannt, ist es nun möglich, Dritten nachzuweisen dass die Nachricht usprünglich auch vom angegebenen Sender stammt. Außenstehende können $F(s)$ berechnen und das Ergebnis mit dem früher veröffentlichten c vergleichen. Stimmt das Ergebnis überein ist sichergestellt, dass der Sender s gewählt hat. F wurde gerade so gewählt,dass das Ergebnis eindeutig ist. Damit ist die Einweg-Funktion F eine Verbindlichkeitsfunktion [PeTy03].

3.2.2 Einweg-Kette

Um eine Einweg-Kette $s_0, ..., s_l$ der Länge l zu erstellen, nimmt man einen Zufallswert für s_l, wendet eine Commitment-Funktion F auf diesen mehrmals an, und erhält dadurch die restlichen Elemente der Kette. Jedes dieser Elemente berechnet sich mit $F^i(s_l) = s_{l-i}$. Zuerst wird $F^1(s_l) = s_{l-1}$ berechnet bis man schließlich $F^l(s_l) = s_0$ erhält.

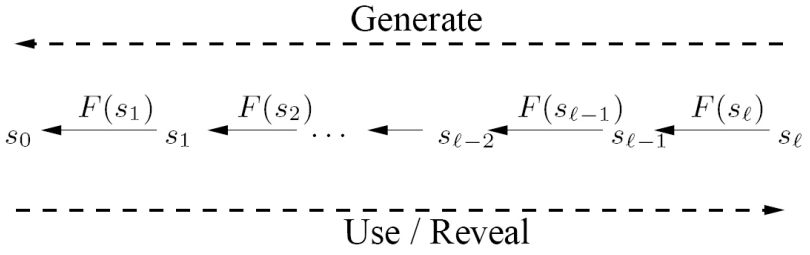

Abbildung 2: Beispiel einer Einweg-Kette [PeTy03]

In Abbildung 2 wird der Aufbau der Kette anschaulich dargestellt. Die Sender veröffentlichen die Werte der Kette immer in der Reihenfolge $s_0, ..., s_l$. Dafür ist es nötig die Kette in umgekehrter Reihenfolge zu generieren.

Der Wert s_0 ist schließlich ein Commitment zur ganzen Einweg-Kette, da mit s_0 überprüft werden kann, ob ein Element in der Kette liegt. Um ein beliebiges s_i zu verifizieren, berechnet man $F^i(s_i) = s_0$. Sollten einige Zwischenwerte verloren gehen, kann der Empfänger diese sogar wiederherstellen, wenn er im Besitz späterer Werte ist, indem er sie neuberechnet [PeTy03].

3.2.3 Selbstzertifizierende Schlüssel

Mit selbstzertifizierenden Schlüsseln kann der Empfänger Nachrichten authentifizieren, ohne zusätzliche Informationen besitzen zu müssen. Es wird somit kein separater MAC für jeden Wert benötigt und Speicherplatz eingespart. Für TESLA werden Commitment-Funktionen für diesen Zweck benutzt. Besorgt sich der Empfänger ein Commitment c zu einem geeigneten Wert s über einen authentifizierten Kanal, ist s ein selbstzertifizierender Schlüssel, da

der Empfänger sofort überprüfen kann ob $F(s) = c$ ist und somit s authentifiziert. TESLA verwendet Einweg-Ketten um eine Folge von selbstzertifizierenden Werten zu erstellen [PeTy03].

3.3 Funktionsweise von TESLA

Da TESLA die Zeit für die benötigte Asymmetrie benutzt, müssen zunächst alle Empfänger mit dem Sender synchronisiert werden. Dabei muss die genaue Differenz zwischen Sender- und Empfängeruhr nicht bekannt sein. Eine Abschätzung Δ, die größer als die tatsächliche Differenz der beiden Uhren ist, genügt.

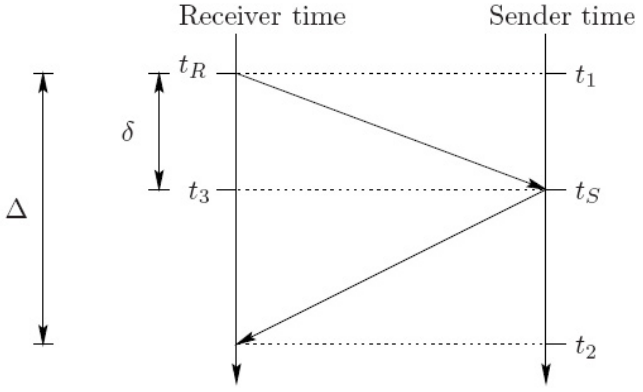

Abbildung 3: Bestimmung des Fehlers Δ [PeTy03]

Abbildung 3 veranschaulicht, wie Δ bestimmt wird. δ ist dabei die Differenz von der Zeit des Empfängers t_R und der des Senders t_S. Δ ist um die Zeit, die ein Paket vom Sender zum Empfänger benötigt, größer und in diesem Beispiel auch die Round-Trip-Time. Dadurch, dass die Empfänger nur eine obere Schranke für die Uhrzeit des Senders wissen müssen, genügen weniger komplexe Algorithmen für die Zeitsynchronisation [PeTy03].

Nun folgt ein Überblick über die einzelnen Schritte bei TESLA.

- Der Sender teilt die Zeit in gleichlange Intervalle ein. Dann erstellt er eine Einweg-Kette von selbstzertifizierenden Schlüsseln und weist jedem Intervall einen Schlüssel zu. Die Einweg-Kette wird dabei so benutzt, dass jeder Schlüssel eines Intervalls genutzt werden kann, um Schlüssel von vorherigen Intervallen daraus zu konstruieren. Zudem legt der Sender eine Verzögerung im Bereich von wenigen Zeitintervallen fest, nach der er die Schlüssel der Einweg-Kette veröffentlicht.
- Der Sender hängt einen MAC an jedes Paket, der über die Inhalte des jeweiligen Paketes berechnet wurde. Um diesen zu berechnen, benutzt er den zum aktuellen Intervall passenden Schlüssel aus der Einweg-Kette. An dieses Paket wird auch der aktuellste Schlüssel, der nach dem Ablauf der Verzögerung veröffentlicht werden darf, angehängt und somit veröffentlicht.
- Jedem Empfänger ist der Zeitplan für die Veröffentlichung von Schlüsseln bekannt. Da zudem die Uhren synchronisiert sind, kann er nachprüfen ob der Schlüssel, welcher benutzt wurde um den MAC zu berechnen, noch geheim ist. Dazu untersucht er, ob der Sender das Intervall in welchem gemäß dem Zeitplan der Schlüssel veröffentlicht werden soll, noch nicht erreicht hat. Ist dies der Fall wird das Paket solange zwischengespeichert, bis der dazu passende Schlüssel veröffentlicht wird.

- Jeder Empfänger überprüft ebenso, ob der Schlüssel, der mit dem Paket veröffentlicht wurde, korrekt ist. Dazu benutzt er die Selbstzertifikation und ältere veröffentliche Schlüssel. Dann überprüft er noch die Korrektheit der MACs von den zwischengespeicherten Paketen, die im dem Intervall gesendet wurden für das der veröffentlichte Schlüssel gilt. Ist der MAC korrekt, wird das Paket akzeptiert.

Im folgenden wird nun detailliert beschrieben, wie Sender und Empfänger schrittweise vorbereitet werden, der Sender Daten überträgt und der Empfänger diese authentifiziert.

3.3.1 Sendersetup

Zuerst bestimmt der Sender die Länge der Zeitintervalle, welche mit T_{int} bezeichnet wird. Die Intervalle berechnen dann wie folgt: $T_i = T_{i-1} + T_{int},\ i = 1, ..., n$. Jeder Schlüssel der Einweg-Kette wird nun der Reihe nach einem Zeitintervall zugewiesen. Dann wird bestimmt, wie lange die Einweg-Kette $K_0, K_1, ..., K_N$ sein soll. Dadurch wird die maximale Sendedauer festlegt bevor eine neue Kette erstellt werden muss. Der Sender wählt nun einen Zufallswert für K_N. Mit einer Einwegfunktion F, die auf einer Pseudo-Zufalls-Funktion basiert, wird der Rest der Kette rekursiv mit der Formel $K_i = F(K_{i+1})$ berechnet. Daraus folgt auch $K_i = F^{N-i}(K_N)$, sodass man jeden Wert der Kette aus K_N berechnen kann, ohne Zwischenwerte zu kennen. Der Schlüssel K_i ist im Intervall i gültig. [PeTy03]

3.3.2 Initialisierung von Empfängern

Zuerst müssen alle Empfänger zeitsynchronisiert sein (siehe Abschnitt 3.3). Für das Initalisierungspaket wird ein gesicherter Übertragungsweg benötigt. Dafür werden Broadcasts oder Unicasts mit digitalen Signaturen benutzt, die einen relativ hohen Berechnungsaufwand haben und somit in Sensornetzen schwierig zu realisieren sind. Der Sender teilt nun den Empfängern folgende Informationen über diesen Übertragungsweg mit:

- Intervalldauer T_{int}, Index des aktuellen Zeitintervalls i, Startzeit T_i des Intervalls i sowie die Länge der Einweg-Kette
- Die Verzögerung 'd' in Anzahl von Intervallen, nach denen ein Schlüssel veröffentlicht wird
- Ein Commitment zur Einweg-Kette

3.3.3 Versand von authentifizierten Nachrichten

Jedes mal, wenn ein Sender eine Nachricht broadcastet, berechnet dieser einen MAC und hängt diesen an die Nachricht an. Zur Berechnung wird der zum passenden Zeitintervall gültige Schlüssel benutzt. Dieser bleibt für die nächsten $d-1$ Intervalle geheim, sodass Nachrichten, die im Intervall i gesendet werden, den Schlüssel K_{i-d} veröffentlichen.

Generell ist es unklug, denselben Schlüssel in mehreren kryptographischen Operationen zu benutzen da es zu Schwächen bei der Verschlüsselung kommen kann [PeTy03]. Daher sollte ein Schlüssel K aus der Einweg-Kette nicht gleichzeitig benutzt werden, um das nächste Element der Kette und den MAC zu berechnen. Deshalb benötigt man eine zweite Einweg-Funktion F', die wie F auf einer Pseudo-Zufalls-Funktion basiert. Mit F' berechnet man den MAC $K'_i = F'(K_i)$. Ein Paket P besteht nun aus folgenden Teilen: $P_i = \{M_i||MAC(K'_i, M_i)||K_{i-d}\}$

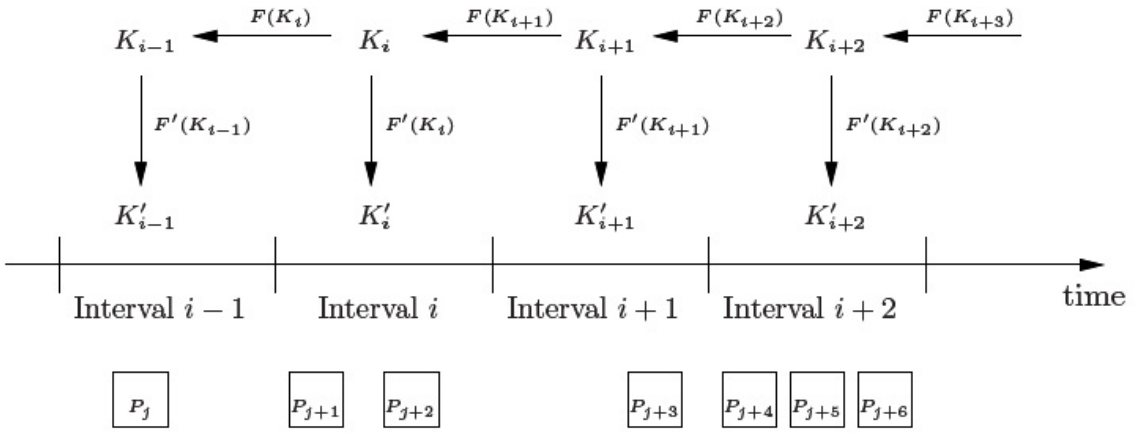

Abbildung 4: Schema von Berechnungen für TESLA [PeTy03]

In Abbildung 4 werden die Berechnungen, die für den Versand eines Paketes notwendig sind, nochmals anschaulich dargestellt. Oben ist die Einweg-Kette, mit deren Elemente durch Anwendung von F' auf K_i der zum Zeitintervall passende MAC K'_i berechnet wird. Unten sind die Pakete dargestellt, die der Sender in den jeweiligen Zeitintervallen verschickt. Für das Paket P_{j+4} zum Beispiel berechnet der Sender einen MAC über die Daten mit dem Schlüssel K'_{i+2}. Nimmt man eine Verzögerung $d = 2$ an, würde dieses Paket auch den Schlüssel K_i beinhalten und veröffentlichen.

3.3.4 Authentifizierung beim Empfänger

Nachdem ein Schlüssel veröffentlicht wurde, haben alle Teilnehmer theoretisch Zugang zu diesem und könnten ein gefälschtes Paket mit einem ebenfalls gefälschten MAC broadcasten. Daher muss jeder Empfänger überprüfen, ob der MAC auf sicheren Schlüsseln basiert und das Paket selbst sicher ist. Sichere Schlüssel sind nur dem Sender bekannt und noch nicht veröffentlicht. Sichere Pakete beinhalten MACs, die mit sicheren Schlüsseln berechnet wurden. Ist das Paket nicht sicher muss es abgelehnt werden, da es gefälscht sein könnte.

Der Sender schickt nun ein Paket P_i, dass wie in Abschnitt 3.3.3 aufgebaut ist, im Intervall i ab. Der Empfänger kann nach dem Erhalt den im Paket P_i enthaltenen selbstzertifizierenden Schlüssel K_{i-d} benutzen, um zu bestimmen in welchem Intervall P_i abgeschickt wurde. Zuerst wird K_{i-d} authentifiziert, zum Beispiel mit dem Schlüssel K_{i-d-2} aus der Einweg-Kette. Gilt $F^2(K_{i-d}) = K_{i-d-2}$ ist der Schlüssel authentisch und wird gespeichert. Nun weiß der Empfänger auch, dass i das Intervall ist, in welchem P_i abgeschickt wurde.

Danach überprüft er mithilfe der Zeitabschätzung Δ das letzte mögliche Intervall x, indem der Sender sich aktuell befinden kann. Ist $x < i + d$ kann das Paket als sicher betrachtet werden, da es nur in diesem Fall auch vom angegeben Sender stammen kann. Der Schlüssel K_i, mit dem das Paket P_i authentifiziert werden kann, wurde aber noch nicht veröffentlicht, da der Sender diesen erst nach der Verzögerung broadcastet. Deshalb muss das Paket solange zwischengespeichert werden, bis K_i beim Empfänger eintrifft.

Erhält der Empfänger nun K_i, überprüft er zuerst ob K_i oder ein neuerer Schlüssel $K_j, j > i$ bereits bekannt ist. Handelt es sich um den aktuellsten Schlüssel, überprüft er die Korrektheit mit einem vorangegangen Schlüssel $K_v, v < i$ und stellt fest, ob er in der Einweg-Kette liegt oder nicht. Ist dies der Fall berechnet er $K'_i = F'(K_i)$ und überprüft somit die Authentizität des Pakets.

Mithilfe eines veröffentlichten Schlüssel lassen sich aufgrund der Eigenschaft der Einweg-Kette alle vergangenen Schlüssel berechnen. Somit hat man die Möglichkeit, bei einem Paketverlust auch sichere, zwischengespeicherte Pakete aus früheren Intervallen zu verifizieren.

Es ist allerdings von Anfang an darauf zu achten, dass die Verzögerung d am Anfang nicht kleiner als die RTT gewählt werden, da sonst die Pakete vom Empfänger verworfen werden. [PeTy03]

3.4 TIK - TESLA mit sofortiger Schlüsselveröffentlichung

Besteht keine Möglichkeit, Daten zwischenzuspeichern, und sind die Uhren von Sender und Empfänger genau synchronisiert, kann man sofortige Authentifizierung ohne die Verzögerung d erreichen. TIK basiert auf der Beobachtung, dass die Verzögerung, die die asymmetrische Eigenschaft garantiert, sich mit der Übertragung einer Nachricht überschneiden kann.

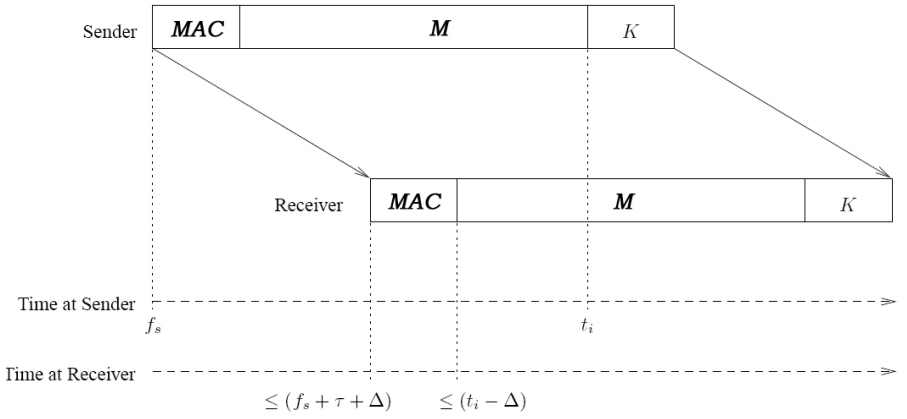

Abbildung 5: Ablaufplan einer TIK Nachricht [HuPJ02b]

In Abbildung 5 bezeichnet f_s die Zeit des Absendens und t_i die Zeit der Schlüsselveröffentlichung. Das Paket besteht aus dem MAC, der Nachricht M und dem Schlüssel K_i, mit dem der MAC berechnet wurde. Wenn c die Ausbreitungsgeschwindigkeit der Wellen ist und r die Reichweite des Senders ist die maximale Übertragungsverzögerung $\tau = \frac{r}{c}$. Daher trifft die Nachricht beim Empfänger spätestens zum Zeitpunkt $(f_s + \tau + \Delta)$ ein. Um mit der Sicherheitsbedingung von TESLA übereinzukommen muss der MAC bei jedem Receiver angekommen sein bevor die Empfängerzeit $t_i - \Delta$ erreicht hat (Δ ist der erlaubte maximale Fehler der Zeitsynchronisation, der vom realen Fehler abweichen kann). Es wird dabei angenommen, dass bei der Übertragung über Funkwellen die Empfangsverzögerungen vernachlässigbar klein ist.

Die Intervalllänge (nicht in der Abbildung eingezeichnet) und die minimale Paketgröße muss so gewählt werden, dass eine Intervallgrenze irgendwo im Paket innerhalb MAC und M existiert. So wird garantiert, dass der Schlüssel im folgenden Intervall veröffentlicht wird.

Das Paket besteht dabei wie in Abbildung 5 zu sehen aus dem MAC, der Nachricht und einem Schlüssel. Nachdem der MAC erhalten wurde wird zuerst überprüft, ob der dazu passende Schlüssel K noch nicht veröffentlicht wurde. Das ist durch die Zeitsynchronisation und der Zeit T_i möglich. Wurde der Schlüssel noch nicht übertragen wird er nach Erhalt dann authentifiziert und mit ihm der MAC überprüft. Sind diese Berechnungen erfolgreich wird das Paket akzeptiert.

Durch die Anwendung von TIK werden auch Wormhole Attacken verhindert. Bei solch einem Angriff leitet „ein Angreiferknoten [...] Nachrichten transparent weiter (bzw. mehrere Angreifer tunneln sie durch das Netz), und täusch(en) damit eine tatsächlich nicht existierende

Verbindung vor“ [Hahn]. Die Pakete werden dann an einer anderen Stelle wieder in das Netz eingespeißt. Das kann zum Beispiel genutzt werden um Pakete zu löschen, wenn der Angreifer sein Wormhole so platziert, das alle Pakete durch das Wormhole geroutet werden [HuPJ02a]. Erneut übertragene Pakete von einem Angreifer werden aber abgelehnt wenn TIK angewendet wird, da die dazu passenden Schlüssel schon veröffentlicht wurden und das Paket deshalb vom Empfänger abgelehnt wird.

Die sehr kurzen Zeitintervalle, die bei TIK nötig sind, führen allerdings auch dazu, dass die Einweg-Kette sehr lange und die Authentifizierung zu rechenintensiv wird. Man benötigt deshalb eine effizierente Methode zur Authentifizierung als Einweg-Ketten.

3.4.1 Hash-Bäume

Eine Alternative zur Kettenstruktur sind Hash Bäume, bei denen die Wurzel als Commitment zu allen Werten dient. Diese Bäume eignen sich dann besonders gut, wenn man entweder nur wenige der Commitments oder die Commitments in beliebiger Reihenfolge öffnen will. Die geheimem Werte $s_i,\ 0 \leq i \leq 2^n$ bilden die Blätter des Baumes, aus denen dann zuerst ein Basis-Commitment $F(s_i) = m_i$ berechnet wird um nachher zu verhindern, dass man ein Blatt entschlüsseln kann ohne den Nachbar zu entschlüsseln. Die inneren Knoten m_i, deren Kinder man als m_{left} und m_{right} bezeichnet, werden dann mit einer Einweg-Funktion F wie folgt berechnet: $m = F(m_{left}||m_{right})$. Die Wurzel ist dann ein Commitment zu allen Blättern. Möchte man einen Wert überprüfen wird von jeder Ebene des Baumes ein Wert benötigt. Insgesamt sind das $log(N)$ Werte, wobei N die Anzahl der Blätter ist.

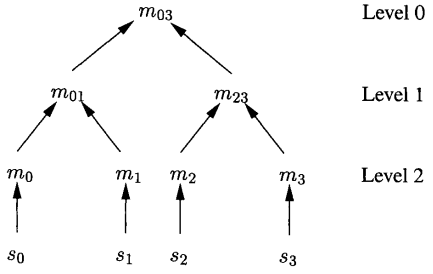

Abbildung 6: Beispiel eines Hashbaums mit Tiefe 3. [PeTy03]

Möchte man das Commitment zu s_2 aus Abbildung 6 öffnen, veröffentlicht man s_2, m_3 und m_{01} Dann wird überprüft ob m_{03} mit $F(m_{01}||F(F(s_2)||m_3))$ übereinstimmt.

4 Nachteile von TESLA und TIK in Sensornetzen

Für Kleinstrechner, wie sie in Sensornetzen eingesetzt werden, ist auch das effiziente TESLA Protokoll nur bedingt geeignet. TESLA authentifiziert das erste Paket, dass das Commitment enthält, mit einer digitalen Signatur. Diese ist zu teuer und aufwändig für Sensorknoten, da der Speicherplatz für lange Schlüssel teilweise nicht ausreicht [PeTy03]. Der Overhead von TESLA liegt bei Anwendung der vorgeschlagenen Verschlüsselungsverfahren bei ungefähr 24 Bytes pro Paket. Das ist bei Sensorknoten, deren Pakete teilweise nur 30 Byte lang sind, ein zu hoher Anteil. Ein weiteres Problem ist die Länge der Einweg-Kette, für die oftmals der Speicherplatz nicht ausreicht. [PeTy03]

5 μTESLA

Aufgrund der in Abschnitt 4 genannten Nachteile wurde μTESLA, eine Erweiterung von TESLA speziell für Sensornetze, entwickelt. Die Reihenfolge der einzelnen Schritte ist dabei genau gleich wie bei TESLA, allerdings wird nun unterschieden, ob nur die Basisstation des Netzes Broadcastet oder die einzelnen Knoten selbst. „Die Basisstation [...] stellt die Verbindung zwischen dem Sensornetz und einem lokalen Netzwerk oder einem Weitbereichsnetz dar“ [fMSu].

Im Gegensatz zu TESLA wird bei μTESLA das Initialpaket nur mit symmetrischen Mechanismen verschlüsselt. Zudem werden Schlüssel statt in jedem Paket nur einmal pro Zeitintervall in einem gesonderten Paket veröffentlicht, Auch die Anzahl der authentifizierten Sender wird eingeschränkt. Das vermeidet, dass zuviele Einweg-Ketten in den Knoten gespeichert werden müssen.

5.1 Detaillierte Beschreibung von μTESLA

Zuerst wird erklärt, wie μTESLA der Basisstation erlaubt, authentifiziert zu Broadcasten.

5.1.1 Sendersetup und Versand von authentifizierten Nachrichten

Wie bei TESLA in Abschnitt 3.3 wird vom Sender mit einer Einweg-Funktion eine Einweg-Kette erstellt. Die Zeit wird wieder in gleiche Intervalle unterteilt und jeder Schlüssel der Einweg-Kette wird mit einem Zeitintervall assoziiert. Zudem wird auch die Zeitspanne d wieder benötigt, nach der Schlüssel veröffentlicht werden. Das Erstellen von MACs funktioniert wie bei TESLA selbst. Mit diesen Informationen können nun Nachrichten versendet werden.

5.1.2 Initialisieren eines neuen Empfängers

Der Empfänger benötigt einen Wert K_i der Einweg-Kette als Verbindlichkeit sowie den Zeitplan der Veröffentlichung von Schlüsseln, der die aktuelle Zeit des Senders T_S, die Startzeit T_i des Intervalls i, die Intervalldauer T_int und die Verzögerung d beinhaltet. Zudem müssen Sender und Empfänger mit einem erlaubten Fehler zeitsynchronisiert sein. Um die Sicherheit der Initialisierung zu erhöhen werden Nonces[3] und MACs eingesetzt. Nonces garantieren dabei die Aktualität der Daten und MACs authentifizieren diese. Für das Berechnen der MACs wird der geheime Schlüssel K_{RS} benutzt, den die Basis und der Knoten von vorneherein miteinander teilen. Die Initalisierung läuft also wie folgt ab, wobei R der Empfänger und S der Sender ist:

$R \rightarrow S : Nonce$

$S \rightarrow R : \ T_S \ || \ K_i \ || \ T_i \ || \ T_{int} \ || \ d \ || \ MAC(K_{RS}, Nonce \ || \ T_S \ || \ K_i \ || \ T_i \ || \ T_{int} \ || \ d)$

Die Daten werden nicht verschlüsselt, da man keine Vertraulichkeit benötigt [PeTy03]. Durch den MAC, der mit dem geteilten Schlüssel und der Nonce berechnet wurde kann der Empfänger die Daten authentifizieren und überprüfen, ob das Paket für die aktuelle Anforderung bestimmt ist. Man benötigt also keine digitale Signatur sondern benutzt den authentifizierten Kanal, den man somit erstellt hat [PeTy03].

[3] Nonces sind Einmalzufallszahlen, welche von den Kommunikationsteilnehmern der Nachricht hinzugefügt werden. Der Partner greift diese auf und verwendet die Zufallszahl für die Antwort. Dadurch weiß der erste Teilnehmer, dass die Nachricht für die aktuelle Sitzung generiert wurde. [BeNS05]

5.1.3 Authentifizierung von Paketen

Nach dem der Empfänger ein Paket erhalten hat, muss die Sicherheit garantiert werden indem überprüft wird, ob der Sender den passenden Schlüssel noch nicht veröffentlicht hat wie in Abschnitt 3.3.4. Sobald der Empfänger einen neuen Schlüssel K_i erhält wird er mithilfe des letzten authentifizierten Schlüssels K_v überprüft. Eine geringe Anzahl von Einweg-Funktionen F wird auf K_i angewandt und man berechnet $K_v = F^{i-v}(K_i)$. Ist die Aussage wahr ist auch der neue Schlüssel authentifiziert und alle Pakete im Intervall v bis i können akzeptiert werden. Dann wird K_i duch K_v ersetzt.

5.2 Broadcasting von authentifizierten Daten durch Sensorknoten

Falls ein Sensorknoten Broadcasten möchte, müssen einige Dinge beachtet werden. Das Speichern von Einweg-Ketten und das Neuberechnen jedes Schlüssels durch den Initalschlüssel K_n ist für Sensorknoten teuer. Zudem könnte der Knoten nicht mit allen potentiellen Empfängern einen Schlüssel teilen. Dann muss der Sender mit diesen Empfängern erst einen teuren Schlüsseltausch durchführen. Der Broadcast von den veröffentlichten Schlüsseln zu allen Empfängern ist ebenfalls teuer und kann wichtige Energie verschwenden.

Es gibt zwei Lösungsansätze für dieses Problem:

1. Der Sensorknoten Broadcastet über die Basisstation. Daten werden zuerst authentifiziert zur Basis übertragen, die danach die Pakete Broadcastet.
2. Der Sensorknoten selbst Broadcastet, die Einweg-Kette wird aber weiterhin von der Basisstation verwaltet, welche dann die Schlüssel nach Bedarf an den Sender schickt. Um Energie zu sparen, kann die Basis zum Beispiel auch die veröffentlichten Keys broadcasten und/oder die Initialisierung für neue Empfänger durchführen.

5.3 Weitere Verbesserungen

Aufgrund der begrenzten Rechen- und Speicherkapazitäten müssen bei μTESLA auch andere Verschlüsselungstechniken als bei TESLA angewandt werden. Durch optimierte Algorithmen wird zum Beispiel der Speicherbedarf nahezu halbiert, wobei natürlich immer ein Konsenz zwischen Schnelligkeit/Speicherbedarf und Sicherheit gefunden werden muss.

Version	Gesamtbedarf	MAC	Verschlüsselung	Schlüsselerstellung
geringster Speicherbedarf	1580	580	402	598
schnellste	1844	728	518	598
Originalversion	2674	1210	802	686

Tabelle 1: Speicherbedarf der Verschlüsselungsverfahren in TinyOS, einem kleinen Betriebssystem das nur 3500 Bytes benötigt [PeTy03]

Tabelle 1 zeigt, wie sich die Standardverfahren noch optimieren lassen. Die genaue Beschreibung dieser optimierten Methoden findet man im Buch [PeTy03].

6 Fazit

Die Vorteile von TESLA und dessen Varianten TIK und μTESLA wurden in dieser Arbeit vorgestellt. Allerdings gibt es auch Nachteile. Die Zeitsynychronisation, die für alle Varianten

benötigt wird, ist in manchen Anwendungsszenarios schwierig zu implementieren. Auch der Schlüsselaustausch zu Beginn ist sehr aufwändig, da die Rechenkapazitäten der Sensorknoten oftmals zu klein für rechen- und speicherintensive, digitale Signaturen ist. Daher muss in Sensornetzen immer auch ein Kompromiss zwischen Sicherheit und Effizienz gefunden werden.

Literatur

[18t07] The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 07). *Shaheen, Ostry, Sivaraman, Jha: CONFIDENTIAL AND SECURE BROADCAST IN WIRELESS SENSOR NETWORKS*, 2007. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04394560.

[Bagc07] Faruk Bagci. Vorlesung Sensornetze, Lehrstuhl für Systemnahe Informatik und Kommunikationssysteme an der Universität Augsburg, 2007. http://www.informatik.uni-augsburg.de/lehrstuehle/sik/lehre/ss/sensornetze/Folien/01_Intro.pdf.

[BeNS05] Beutelspacher, Neumann und Schwarzpaul. *Kryptografie in Theorie und Praxis.* Friedr. Vieweg & Sohn Verlag, Wiesbaden. 2. Auflage, 2005.

[Blaß07] Erik-Oliver Blaß. *Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen.* Dissertation, Universität Karlsruhe (TH), 2007.

[BäMü08] Bäumler und Müller. Schlauer Staub fürs Weingebiet. *sciencegarden - Magazin für junge Forschung*, 2008.

[fMSu] Fraunhofer-Institut für Mikroelektronische Schaltungen und Systeme. Drahtlose Sensornetze. http://www.ims.fraunhofer.de/uploads/media/Wireless_Sensor_Network_d_02.pdf.

[Glob] GlobalSecurity. Sound Surveillance System (SOSUS). http://www.globalsecurity.org/intell/systems/sosus.htm.

[Gola02] Frank Golatowski. Vorlesung Drahtlose Sensornetzwerke, Universität Rostock, 2002. http://www-md.e-technik.uni-rostock.de/ma/gol/lectures/wirlec03/VL2002/wirlec05-snw.ppt.

[Hahn] Björn Hahnenkamp. Secure OLSR - Angriffszenarien und Schutzmechanismen. Seminar Mobiles Internet, WS05/06, Universität Karlsruhe.

[HuPJ] Hu, Perrig und Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. Working Session on Security in Ad Hoc Networks June 12th, 2002, http://www.hit.bme.hu/~buttyan/June12/presentations/Adrian_Perrig.pdf.

[HuPJ02a] Hu, Perrig und Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks, 2002. http://www.monarch.cs.rice.edu/monarch-papers/tikreport.pdf.

[HuPJ02b] Hu, Perrig und Johnson. Wormhole Detection in Wireless Ad Hoc Networks, 2002.

[IIS] Fraunhofer IIS. Was ist ZigBee? http://www.iis.fraunhofer.de/bf/med/sensorik/anw/zigbee/zigbee_was.jsp.

[Kompa] Elektronik Kompendium. *IEEE 802.11 / Technische Grundlagen WLAN.* http://www.elektronik-kompendium.de/sites/net/0907101.htm.

[Kompb] Elektronik Kompendium. *IEEE 802.11 / Wireless LAN (WLAN) / Wireless Fidelity (WiFi).* http://www.elektronik-kompendium.de/sites/net/0610051.htm.

[Krae06] Rolf Kraemer. Sensornetze im medizinischen Umfeld, 2006. http://www.leibniz-institut.de/cms/pdf/Kraemer-Sensornetze_im_medizinischen_Umfeld.pdf.

[Krau] Tobias Krauer. Zeitsynchronisation in Sensornetzen, Seminararbeit. `http://www.vs.inf.ethz.ch/edu/SS2003/DS/reports/05_time_report.pdf`.

[Micr06] Microsoft. STOP 0x0000000A / STOP 0x00000019 aufgrund Teardrop-Angriffe, 2006. `http://support.microsoft.com/kb/179129/de`.

[PeTy03] Adrian Perrig und J.D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers. 2003.

[Socia] IEEE Computer Society. *IEEE Std 802.11-2007*. `http://standards.ieee.org/getieee802/download/802.11-2007.pdf`.

[Socib] IEEE Computer Society. *IEEE Std 802.15.1-2005*. `http://standards.ieee.org/getieee802/download/802.11-2007.pdf`.

[Socic] IEEE Computer Society. *IEEE Std 802.15.4-2006*. `http://standards.ieee.org/getieee802/download/802.11-2007.pdf`.

[Wätj08] Dietmar Wätjen. *Kryptographie Grundlagen, Algorithmen, Protokolle*. Spektrum Akademischer Verlag, Heidelberg. 1. Auflage, 2008.

[Wels05] Matt Welsh. Wireless Communications and Sensor Networks, Vorlesung Universiät Harvard, 2005. `http://www.eecs.harvard.edu/~mdw/course/cs263/notes/intro.pdf`.

Abbildungsverzeichnis

Schlüsselverteilung in Sensornetzen

Simon Lorenz

Diese Ausarbeitung, erstellt im Rahmen des Seminars Netzsicherheit und Hackerabwehr, befasst sich mit den verschiedenen Aspekten der Schlüsselaushandlung und deren Verteilung in drahtlosen Sensornetzen. Da Sensornetze noch ein recht junges Forschungsgebiet darstellen, existiert hierfür noch kein Standard, dafür aber eine Vielzahl von Ansätzen, um die Kommunikation in einem drahtlosen Sensornetz sicherer zu machen. Im Verlauf der Arbeit wird versucht, einen Überblick über verschiedene Ansätze zur Schlüsselverteilung zu gegeben. Wie dies in herkömmlichen Netzen geschieht und wesshalb herkömmliche Verfahren für Sensornetze unbefriedigende Ergebnisse liefern. Anschließend werden zwei Verfahren im Detail behandelt, die auf unterschiedlichen Modellen beruhen.

1 Einleitung

Im Zuge der voranschreitenden Entwicklung hin zur drahtlosen Kommunikation, eröffnen sich zahlreiche neue Möglichkeiten und Anwendungsgebiete. Eine vielversprechende und möglicherweise zukunftsweisende Technologie stellen drahtlose Sensornetzwerke dar. Durch ihren flexiblen verteilten Ansatz, sind sie in der Lage in Umgebungen zu operieren, die zuvor nicht denkbar gewesen sind.

1.1 Hintergrund

Sensornetze werden in der Regel zur Überwachung und Analyse der Umgebung eingesetzt. Die Anwendungsmöglichkeiten dieser Netze sind enorm. Sinnvolle Anwendungsbeispiele, um nur ein paar zu nennen, sind z.B die frühzeitige Erkennung von Erdbeben, Waldbränden und anderen Naturkatastrophen wie auch die Überwachung von Naturschutzgebieten oder sonstigen Lebensräumen bedrohter Tierarten. Auch die periodische Analyse von Gebäuden und das frühzeitige erkennen einsturzgefährdeter Gebäude könnte ungemein zur Bausicherheit beitragen. Die medizinische Betreuung von Patienten an ihrem Wohnort mittels Sensornetzen, könnte die Reaktionszeit bei einem Notfall signifikant verbessern. Auch auf militärischem Gebiet existieren viele denkbare Szenarien.

Sensornetze bestehen aus einer meist großen Anzahl von batteriebetriebenen Kleinstcomputern, die mit Komponenten ausgestattet sind, die es ihnen erlauben ihre Umgebung zu einem gewissen Ausmaß abzutasten bzw. Veränderungen in ihr zu messen, Daten zu verarbeiten und Daten über ein drahtloses Netzwerk zu kommunizieren. Diese Geräte werden nachfolgend Sensorknoten genannt. Ein wesentlicher Vorteil bei Sensorknoten besteht darin, dass mit ihrer Hilfe viele detaillierte Informationen über ein beobachtetes Gebiet gesammelt und bereits im Sensornetzwerk zu einem gewissen Grad aufbereitet werden können, was große Rohdatenströme vermeidet und den Abruf bereits aggregierter Daten ermöglicht[PeSW04].

Die sich eröffnenden Möglichkeiten bringen neue Herausforderungen und Probleme mit sich. Wie schützt man diese Daten vor unbefugten Dritten und böswilligen Angreifern? Die Probleme bei Sensornetzen zur Sicherheit sind insofern speziell, da erprobte Verfahren, die in traditionellen Netzwerken Anwendung finden, hier zu keinem befriedigendem Ergebnis führen. Sensorknoten haben zu beschränkte Ressourcen, um konventionelle Verfahren zur Datensicherheit bewerkstelligen zu können. Generell gilt, um Daten vor dem Zugriff Dritter zu schützen, dass, noch bevor die Daten verschlüsselt werden, die Kommunikationspartner (vorliegend die Sensorknoten) sich gegenseitig authentifizieren müssen. Erfolgt diese Authentifikation nicht, ist auch eine anschließende verschlüsselte Übertragung der Daten wertlos, da nicht gewährleistet ist, an wen übertragen wird. Für Authentifikation wie auch für die Verschlüsselung von Daten ist, zumindest bei symmetrischen Verfahren, ein gemeinsames Geheimnis nötig - nachfolgend mit Schlüssel bezeichnet - unter dessen Zuhilfenahme die Daten chiffriert werden.

1.2 Zielsetzung

Diese Arbeit befasst sich mit verschiedenen Ansätzen zur Verteilung und Aushandlung von Schlüsseln in Sensornetzen. Hierzu werden im ersten Teil die Probleme beim Einsatz von Sensornetzen und die daraus resultierende spezielle Sicherheitsthematik erläutert. Anschließend wird auf die Frage eingegangen, weshalb konventionelle Schlüsselverteilungsmechanismen in Sensornetzen zu schlechten Ergebnissen führen. Darüber hinaus wird ein Überblick zu speziellen Schlüsselverteilungsverfahren in Sensornetzen gegeben, von denen LEAP und das EG-Schema mit Modifikationen, detailliert beschrieben, behandelt und gegenübergestellt werden.

2 Grundlagen

Sensorknoten bestehen aus mehreren unterschiedlichen Komponenten. Diese sind ein Prozessor, ein Datenspeicher, ein oder mehreren Sensoren und ein Transmitter für drahtlose Kommunikation. Diese Komponenten, die nicht zwingend einzeln zusammen geschaltet sein müssen, sondern auch auf einem einzigen Computerchip implementiert werden können, werden von einer Batterie angetrieben.

2.1 Allgemeine Probleme bei Sensornetzen

Durch die Zielvorgabe, ein möglichst kleines Gerät herzustellen, sind Sensorknoten sehr starken Limitierungen unterworfen. Zu diesen zählen eine beschränkte Rechen- sowie Datenpeicherkapazität und eine begrenzte Lebensdauer bzw. Laufzeit. So haben auch derzeitige neuere Geräte meist lediglich ein Speichervolumen von 512 kB und RAM-Größen von wenigen Kilobyte [Xbow08]. Zu diesen Ressourcenlimitierungen kommt erschwerend hinzu, dass der Kommunikationsoverhead bei drahtlosen Netzen wesentlich höher als bei drahtgebundenen Netzen ist. Es werden mehr Steuer- und Kontrolldaten benötigt, um Nutzdaten sicher und ohne Integritätsverlust zum Ziel zu übertragen. Es besteht ebenfalls ein Skalierbarkeitsproblem, da Sensornetze für möglichst detailgenaue Informationen über die Umgebung zum einen, an Anzahl der Sensorknoten sehr hoch sein sollten, und zum anderen, im eingesetzen Gebiet auch sehr dicht verteilt sein können. Hierbei ist auch zu beachten, dass die genaue Verteilung der Sensorknoten vor ihrer Allokation im Zielgebiet nicht vorher bekannt ist, d.h. es existieren keine Informationen bezüglich der Netzwerk-Topologie und somit auch keine Informationen, welche Nachbarknoten sich in Kommunikationsreichweite eines Knotens befinden und wie

diese positioniert sind [CaYe05]. Desweiteren besteht ein hohes Risiko, dass Sensorknoten physisch angegriffen werden. Normale Computer sind meist unter der physischen Kontrolle des Eigentümers und einem Dritten kann der Zugang leicht verwehrt werden. Unbeaufsichtigte Sensorknoten sind klein genug, um sie einfach für die spätere Analyse einzusammeln. All diese Herausforderungen machen eine sichere Kommunikation und daraus folgend eine effiziente Schlüsselverteilung im Sensornetzwerk umso wichtiger.

2.2 Sicherheitsziele

Vor Umsetzung und Anwendung eines Schlüsselverteilungsverfahrens ist es sinnvoll, Sicherheitsziele die erreicht werden sollen, zu definieren. Erst anhand dieser ist eine spätere Evaluation eines Verfahrens auf dessen Tauglichkeit überhaupt möglich.

2.2.1 Allgemeine Sicherheitsziele

Allgemeine Sicherheitsziele bei der Kommunikation und ihre Umsetzung in *herkömmlichen* Netzwerken sind.

Vertraulichkeit Die Geheimhaltung der übertragenen Daten ist notwendig. Daten sollen nicht für Dritte einsehbar sein. Realisiert über langsame asymmetrische Verschlüsselungsverfahren (RSA, El-Gamal) als auch über schnellere symmetrische Verfahren (AES, DES, RC-6)

Authentizität Beschreibt die Echtheit und Glaubwürdigkeit der Kommunikationspartner. Es wird ein Identitätsnachweis gefordert. Dieser erfolgt in der Regel über ein Zertifikat, das den öffentlichen Schlüssel, Name des Inhabers (zusätzl. weitere Informationen) und die Signatur einer vertrauenwürdigen Stelle (CA) enthält. Alternativ kann auch über ein Challenge-Response Verfahren oder über einen Message Authentication Code (MAC) authentifiziert werden. Dabei ist zu beachten, dass das Geheimnis dann aber bereits vorverteilt sein muss.

Integrität Daten sollen korrekt sein. Falls sie das nicht sind, so muss erkennbar sein, ob sie manipuliert wurden. Eine Prüfsumme (CRC) oder ähnliches reicht nicht aus. Daten müssen mit dem privaten Schlüssel des Senders signiert werden (in der Praxis oft nur der Hash-Wert dieser) um die Integrität sicherzustellen.

Autorisation Es muss überprüft werden, ob ein bereits authentifizierter Kommunikationspartner auch die Berechtigung hat, um auf Daten zuzugreifen. Dies wird in der Regel über Access-Control-Lists (ACLs) oder Attribut-Zertifikate realisiert.

Verbindlichkeit Eine verschickte Nachricht und ihr Inhalt sollen für den Sender unabstreitbar sein. Der Sender darf sich nicht darauf berufen können, die Nachricht nicht geschickt zu haben. Dies wird ebenfalls über eine Signatur realisiert.

Verfügbarkeit Möglichst keine bzw. nur eine geringe zeitliche Einschränkung bei der Abrufbarkeit von Daten soll erreicht werden. Realisiert durch DDoS-Gegenmaßnahmen (Puzzles, Cookies) und physisch realisiert durch Akkus für unterbrechungsfreie Stromversorgung (USVs).

2.2.2 Relevante und spezielle Sicherheitsziele bei Sensornetzen

Einige der oben genannten Ziele sind in Sensornetzen relevanter als andere. Mechansimen zur Autorisation und die zur Verbindlichkeitkeit von Nachrichten können sicherlich in Sensornetze implementiert werden, der Schwerpunkt dieser Ausarbeitung liegt jedoch auf Authentifikation,

Vertraulichkeit und Integrität. Das Ziel der Verfügbarkeit ist ebenfalls ein wichtiges Kriterium, aus diesem leitet sich die Überlebensfähigkeit eines Sensorknotens ab. Ein weiterer wichtiger Aspekt ist das Sicherheitslevel der Kommunikation, da nicht zwingend alle übertragenenen Daten gleich sensibel sind. So ist es beispielsweise gut, wenn man Steuerinformationen wie Paketköpfe mit verschlüsseln kann, wichtiger jedoch ist, dass der Payload, also die Nutzdaten, immer so gut gesichert werden, wie es die momentane Situation erlaubt.

Überlebensfähigkeit Stellt die Fähigkeit dar, gewisse Dienste auch im Falle von Energieverlusten, Beschädigungen am Sensorknoten oder Angriffen, bereitzustellen. Da Sensornetze auch in unwirtlichen Umgebungen eingesetzt werden sollen, ist eine gewisse Robustheit gegen Umwelteinflüsse oder physische Angriffe notwendig.

Qualität der Sicherheitsdienstgüte - QoSS Das Sicherheitsniveau, das bei der Kommunikation eingehalten werden soll, sollte anpassbar sein. Je niedirger das Sicherheitsniveau, desto geringer der Berechnungsaufwand. Dies stellt sicher, dass auch bei einem Leistungsengpass Kommunikation noch stattfinden kann.

Widerstandsfähigkeit gegen Schlüsselkompromittierung Sofern ein Sensorknoten von einem böswilligen Dritten entwendet wurde und jener es erreicht hat, den Schlüssel des Knotens zu extrahieren, ist es wichtig, dass er keine, bzw. so wenig wie möglich an Sicherheitsinformationen von anderen Sensorknoten preisgibt bzw. preisgeben kann.

2.3 Anforderungskriterien an die Schlüsselverteilung

Um die oben genannten Sicherheitsziele in Sensornetzen einhalten zu können, muss das Verfahren zu Schlüsselverteilung bestimmte Anforderungen erfüllen. Die wichtigsten sind:

Skalierbarkeit Der Verteilungsmechanismus sollte auch noch bei sehr großen Netzen funktionieren. Dies erstreckt sich auch auf das Hinzufügen vieler Knoten bei einem bereits allokierten und aufgebautem Sensornetzwerk.

Effizienz Die Ressourcenbeschränkungen von Sensorknoten müssen beachtet werden. Die Verteilung der Schlüssel sollte minimal sein bzw. einen guten Trade-Off finden bezüglich dem Speicheraufwand, dem Rechenaufwand und auch dem benötigten Kommunikationsaufwand. Der Kommunikationsaufwand ist nicht zu vernachlässigen, da durch bessere Hardware auf den Sensorknoten zwar Speicher, Berechnungskapazität sowie auch die Laufzeit verbessert werden kann, nicht jedoch die Anzahl der zu übertragenden Nachrichten; dies ist größtenteils protokollabhängig. Da der Energieverbrauch proportional zum Quadrat der Kommunikationsdistanz ist [EsV.02], besteht somit auch hier ein Sparsamkeitsgebot.

Konnektivität der Schlüssel Stellt die Wahrscheinlichkeit dar, mit der zwei oder mehr Sensorknoten einen gemeinsamen Schlüssel besitzen, über den sie kommunizieren können. Hierauf wird bei einem den vorgestellten Verfahren noch näher eingegangen.

2.4 Netzwerkmodelle

Sensornetzwerke können, wie jedes andere Netzwerk auch, hierarchisch oder verteilt betrieben werden. In einem hierarchischen Netzwerk kommen Basis-Stationen zum Einsatz. Basistationen haben wesentlich mehr Ressourcen als herkömmliche Knoten zur Verfügung und sind üblicherweise Verbindungspunkt zu einem oder mehreren anderen Netzwerken. Über sie kann ein Großteil der Kommunikation stattfinden und sie können zur Schlüsselverteilung genutzt werden, indem die Basisstation als Key-Distribution-Center (KDC) verwendet wird. Sie vereinfachen zwar die Schlüsselverteilung, haben dafür aber den Nachteil, dass sie einen zentralen

Angriffspunkt im Netzwerk darstellen. Sind eine oder mehrere Basisstationen einmal kompromittiert, so fallen auch alle die von ihr abhängigen Sensorknoten aus. Ein anderer Ansatz ist das verteilte Netzwerk. Sensorknoten müssen Schlüssel jetzt untereinander aushandeln und es existiert keine zentrale Stelle zur Datenaggregation. Dies macht die Schlüsselverteilung komplizierter und berechnungsaufwändiger für die Sensorknoten, hat aber den Vorteil, dass eine hohe Streung des Risikos erreicht wird und das Netzwerk weniger angreifbar ist. Im späteren Verlauf werden Verfahren vorgestellt, von denen einige auf hierarchischen Netzen beruhen und andere auf verteilten Netzen.

3 Schlüsselaushandlungs und -verteilungsmechanismen

Für die Aushandlung und Verteilung von Schlüsseln existiert eine Vielzahl von Verfahren. Mit all diesen Verfahren gehen bestimmte Vor- und Nachteile einher. Warum Sensornetze spezielle Verfahren benötigen, soll nachfolgend deutlich gemacht werden.

3.1 Diffie-Hellman und die Probleme

Das gängigste Verfahren zur Schlüsselaushandlung in traditionellen Netzen stellt das Verfahren von Diffie-Hellman dar. Bei diesem Verfahren wird ein gemeinsamer Schlüssel ausgehandelt, bei dem alle Nachrichten, die zur Aushandlung des Schlüssel verschickt werden müssen, auch von Dritten eingesehen werden können, und es diesen nicht erlaubt, Rückschlüsse auf den gemeinsamen Schlüssel zu ziehen. Das Verfahren wird üblicherweise eingesetzt, um mit dem so ausgehandelten Schlüssel später über ein symmetrisches Krypto-Verfahren zu kommunizieren. Bei asymtrischen Verfahren kann der geheime und der öffentliche Schlüssel lokal erzeugt werden, wobei der öffentliche Schlüssel jeweils verschickt wird. Es besteht bei asymmetrischen Verfahren keine Notwendigkeit, ein gemeinsames Geheimnis zu vereinbaren, es muss lediglich der zum privaten Schlüssel inverse öffentliche Schlüssel dem Kommuniktionspartner zugänglich gemacht werden.

Ferner bietet das Diffie-Hellman Verfahren Perfect Forward Secrecy, d.h., dass ein kompromittierter Schlüssel keine Informationen über bereits zuvor ausgehandelte Schlüssel enthält. Jeder Schlüssel wird individuell neu vereinbart. Das Verfahren funktioniert so, dass in einer endlichen zyklischen Gruppe (nachfolgend G genannt) ein schwieriges Dlog-Problem realisiert wird. Es ist klar, dass die Lösung des diskreten Logarithmusproblems erlaubt, das Diffie-Hellman-Problem zu lösen. Die Umkehrung ist nicht bekannt. Weiterhin wird ein Erzeuger bzw. die Basis der Gruppe (nachfolgend g genannt) und eine beliebige Primzahl (p) frei gewählt [Resc99]. Die Kommunikationspartner werden mit Knoten 1 und Koten 2 bezeichnet, der ausgehandelte Schlüssel mit Key.

- Knoten 1 wählt g und p frei. Knoten 1 berechnet $K_1 = g^a \ mod \ p$ in G, wobei die Zufallszahl a von Knoten 1 frei gewählt ist. Knoten 1 schickt K_1, g und p an Knoten 2.
- Knoten 2 wählt eine Zufallszahl b und berechnet $K_2 = g^b \ mod \ p$ in G und schickt K_2 an Knoten 1
- Knoten 2 berechnet $Key = K_1^b \ mod \ p$ in G
- Knoten 1 berechnet $Key = K_2^a \ mod \ p$ in G

Es kann leicht gezeigt werden, dass Knoten 2 und Knoten 1 denselben Key berechnen.

Knoten 2: $Key = K_1^b \ mod \ p = (g^a \ mod \ p)^b \ mod \ p = g^{ab} \ mod \ p$
Knoten 1: $Key = K_2^a \ mod \ p = (g^b \ mod \ p)^a \ mod \ p = g^{ba} \ mod \ p = g^{ab} \ mod \ p$

Der so ausgehandelte Schlüssel ist für einen Dritten, auch wenn er alle Nachrichten einsehen konnte, nicht ermittelbar, da er das Geheimnis von Knoten 1 oder Knoten 2, d.h. die jeweilige Zufallszahl mit der potenziert wurde, nicht kennt, da diese nicht übertragen wurden. Für Sensornetze besteht das Problem, dass das potenzieren in endlichen zyklischen Gruppen berechnungsintensiv ist und Sensorknoten der momentanen Bauart entweder damit noch überfordert sind oder die Effizienz des Sensornetzwerks stark verringert wird. Es gibt jedoch spezielle Ansätze für Sensornetze wie IKA2 [StTW00] und das Schema von Burmester und Desmedt [BuDe94], welche auf dem Diffie-Hellman Verfahren beruhen und versuchen, die Effizienz zu erhöhen. Jedoch liefern diese Ansätze auch insofern unbefriedigende Ergebnisse, dass Authentifikation bei Diffie-Hellman ein Problem darstellt ist. Das Diffie-Hellman Verfahren hat den Nachteil, dass es Man-in-the-Middle angreifbar ist. So könnte im Beispiel oben ein zuvor modifizierter Knoten sich als einer der beiden kommunizierenden Knoten ausgeben, indem er die initiale Nachricht mit den Werten K_1, g und p abfängt, mit einer von ihm frei gewählten Zufallszahl anschließend K_{2fake} berechnet und dies an Knoten 1 schickt. In traditionellen Computernetzen wird dieses Problem durch Zertifikate gelöst, die von einer vertrauenswürdigen Instanz (Root-CA) ausgestellt werden. Diese CA wird in der Regel für das Sensornetz nicht erreichbar sein, ein Internet-Gateway ist aufgrund der vorher noch nicht bekannten Einsatzumgebung nicht zwingend gegeben. Für Sensornetze müsste also eine Zertifizierungsstelle implementiert werden, bei der die Identität von Knoten überprüft werden könnte. Dieser Ansatz würde sich jedoch nur für hierarchische Netze eignen. Es kommt hinzu, dass auch ein Zertifikatsstandard für Strukturierung von Zertifikaten, wie X.509 für herkömmliche Netze, erst noch geschaffen werden müsste. Es existieren jedoch auch neuere Ansätze für hierarchische Netze und dem Einsatz von Public-Key Kryptographie, die sich mit dem Authentifikationsproblem befassen [NAMR+08]. Normalerweise wird bei den meisten Verfahren die Authentifikation der Knoten über Schlüssel-IDs gelöst wird, die bei der Basisstation oder in den Sensorknoten selbst gespeichert werden, bevor diese allokiert werden, und diese damit dann gültige Schlüssel verifizieren können.

3.2 Intuitive Ansätze

Da Effizienz und Sicherheit (s.o. Kap. 2.2 Sicherheitsziele) sich entgegenstehende Ansprüche im Umfeld von Sensornetzen sind, wäre ein erster intuitiver Ansatz, dass Sensorknoten nicht nach ihrer Allokation erst Schlüssel aushandeln müssen, sondern einen oder mehrere Schlüssel - bevor die Positionierung am Bestimmmungsort erfolgt - bereits zugewiesen bekommen. Die einfachste Lösung ist hierbei sicherlich die eines netzwerkweiten gemeinsamen Schlüssels (nachfolgend Master-Key gennant) für alle Sensorknoten. Diese Lösung würde zwar gut skalieren, jedoch wäre sie in höchstem Maße unsicher, da die Wiederstandsfähigkeit gegenüber Schlüsselkompromietierung (Kap. 2.2.2) gleich null ist. Ist ein Sensorknoten in die Hände eines Dritten geraten, würde dies ihm erlauben, den Netzwerkverkehr aller Knoten zu entschlüsseln. Es existieren Implementierungsversuche wie BROSK (Broadcast Session Key Negotiation Protocol), die diesen Ansatz verfolgen und die Sicherheit durch den Austausch von Zufallszahlen zu erhöhen versuchen. Trotzdem kann das Verfahren als unsicher angesehen werden, da die Verbesserungen nichts an der Tatsache ändern, dass sofern der Master-Key einmal ermittelt wurde, jede Nachricht im Netzwerk entschlüsselt werden kann. Eine Variante dieser Idee ist, einen Master-Key zu benutzen, über diesen dann die Knoten paarweise Schlüssel unter sich aushandeln. Ist diese Aushandlung erfolgt, wird auf alle Knoten der Master-Key gelöscht. Diese Verfahren hat zum einen den Nachteil, dass es bevor der Master-Key gelöscht wurde, eine hohe Angreifbarkeit hat und es zum anderen so unmöglich ist, zusätzliche Sensorknoten nach der Allokation des Sensornetzes hinzuzufügen.

Das Gegenextrem zu einem Master-Key auf allen Sensorknoten wäre allen Sensorknoten

einen jeweils paarweisen Schlüssel untereinander zuzuweisen. Wenn jeder Knoten mit jedem Knoten im Netz einen eigenen Schlüssel zugewiesen bekommen würde, folgt daraus, dass jeder Knoten bei n Knoten im Netz $n-1$ Schlüssel speichern müsste. Ferner müssten $n * \frac{(n-1)}{2}$ Schlüssel im Netzwerk verteilt werden. Dieser Ansatz bietet zwar eine hohe Widerstandsfähigkeit gegen Schlüsselkompromitierung, skaliert dafür aber sehr schlecht. Auch wenn der hohe Speicheraufwand seitens der Knoten in Kauf genommen wird, so wird trotzdem Speicherplatz verschwendet, da Knoten schon aufgrund ihrer begrenzten Kommunikationsreichweite nicht mit allen Knoten im Netz kommunizieren können, bzw. dies nur über andere Knoten realisieren können die als Hop dienen. Es würden also niemals alle verteilten Schlüssel zum Einsatz kommen.

3.3 Verfahren basierend auf hierarchischen Netzen - LEAP

Ein Verfahren, einen guten Trade-Off zwischen Effizienz und Sicherheit zu finden, stellt LEAP (Localized Encryption and Authentication Protocol) dar. LEAP baut auf einem hierachischen Netz mit Basisstationen auf. Bei LEAP wird von der Annahme ausgegangen, dass Sensorknoten nach ihrer Allokation nicht mobil sind, desweiteren Basisstationen existieren und diese nicht kompromittiert werden können. Da drahtlose Kommunikation ein besonderes Sicherheitsrisiko darstellt, wird davon ausgegangen, dass ein Angreifer Nachrichten abfangen kann, Pakete einfügen und alte von Knoten verschickte Nachrichten noch einmal (Replay) schicken kann. Ferner wird davon ausgegangen, dass alle Daten eines kompromittierten Knotens für einen Angreifer einsehbar sind. Zu den bereits erläuterten Sicherheitszielen kommen bei LEAP die Unterstützung für Verschiedene Kommunikationsarten wie Unicast, lokaler und globaler Broadcast als Designziel hinzu [ZhSJ04].

Das Besondere an LEAP ist, dass es verschiedene Schlüssel zu verschiedenen Zwecken zur Verfügung stellt. D.h., dass die Dienstgüte (QoSS) und das Sicherheitsnivau angepasst werden kann. Die vier Schlüsselarten sind: Ein individueller Schlüssel zur Kommunikation mit der Basisstation, ein paarweiser Schlüssel zur Kommunikation mit einem anderen Sensorknoten, einen gemeinsamen Clusterschlüssel zur Kommunikation mit mehreren benachbarten Knoten und einen Gruppenschlüssel zur Kommunikation mit allen Knoten im Netzwerk. Das Verfahren für die Schlüsselverteilung ist je nach Schlüssel unterschiedlich.

Individueller Schlüssel Dieser Schlüssel wird ausschließlich für die Kommunikation mit der Basisstation genutzt. So kann z.B. ein Sensorknoten seinen individuellen Schlüssel dazu benutzen einen Message Authentication Code (MAC) von aufgezeichneten Daten zu berechnen, sofern die Daten von der Basisstation verfiziert werden sollen. Die Basisstation kann den individuellen Schlüssel dazu benutzen, spezielle Anweisungen an einen bestimmten Sensorknoten zu senden, oder sensible Daten wie Schlüsselmaterial für die Kommunikation mit benachbarten Knoten damit zu verschlüsseln

Paarweiser Schlüssel Jeder Knoten teilt sich einen paarweisen symmetrischen Schlüssel mit seinen unmittelbaren Nachbarn. Die paarweisen Schlüssel werden benutzt um Daten abzusichern, die Vertraulichkeit oder Authentifikation erfordern. So kann ein Sensorknoten z.B den paarweisen Schlüssel dazu benutzen, gemessene Daten an einen Nachbarknoten zu übertragen, der die Daten aggregiert.

Clusterschlüssel Der Clusterschlüssel wird von einem Knoten und allen seinen Nachbarn geteilt. Er wird hauptsächlich dazu benutzt, um lokale Broadcast-Nachrichten zu sichern wie z.B Routing Informationen. Desweiteren ermöglicht er die passive Anteilnahme anderer Knoten an der Kommunikation. So kann ein Knoten, der die Daten eines anderen Knoten mitlesen kann, obwohl diese nicht explizit für ihn bestimmt sind, davon insofern profitieren, dass er entscheiden kann, ob er seine gemessenen Daten verschicken

soll oder nicht, da es die gleichen sein könnten und somit kein zusätzlicher Mehrwert an Information entsteht. Der Kommunikationsaufwand kann somit reduziert werden, was die Laufzeit des Knotens erhöht.

Gruppenschlüssel Dieser Schlüssel wird von der Basisstation genutzt, um Nachrichten zu verschlüsseln die per globalem Broadcast an jeden Sensorknoten im Netz adressiert sind, wie z.B. Anweisungen, die alle Knoten betreffen. Hierbei ist jedoch zu beachten, dass ein effizienter Rekeying-Mechanismus zur Verteilung eines neuen Gruppenschlüssels benötigt wird, für den Fall, dass der Gruppenschlüssel durch einen kompromittierten Knoten offen gelegt wurde.

Verteilung Individueller Schlüssel — Der Individuelle Schlüssel, den jeder Knoten mit der Basisstation besitzt, wird vor der Allokation der Sensorknoten auf diesen gespeichert, da dies in einer kontrollierten Umgebung stattfinden kann. Der individuelle Schlüssel K_u^m für jeden Knoten u wird durch eine Pseudozufallsfunktion generiert. f_k bezeichne die Pseudozufallsfunktion und K^m den Master-Key der Basisstation, den nur sie kennt. Dabei wird die Pseudozufallsfuntion mit der eindeutigen ID des Knoten u und dem Master-Key des Basisstation parametrisiert. Die ID-Liste, die die Relation zwischen IDs und den jeweiligen Knoten herstellt, ist auf der Basisstation gespeichert. Die Berechnung hat also die Form $K_u^m = f_{K^m}(u)$ Diese Vorgehensweise hat den Vorteil, dass die Basisstation für die einzelnen paarweisen Schlüssel keine Zustandshaltung betreiben bzw. diese speichern muss. Wenn sie mit einem bestimmten Knoten kommunizieren will, berechnet sie den individuellen Schlüssel für den Knoten direkt (on-the-fly), da sie die Paramter (Master-Key und Knoten-ID) kennt.

Verteilung paarweiser gemeinsamer Schlüssel — Diese Schlüssel werden nur zwischen einem Knoten und seinen unmittelbaren Nachbarn ausgehandelt. Auch hier erfolgt eine Vorverteilung eines initialen Schlüssels K_I zu den Knoten noch vor ihrer Allokation. Aus diesem Schlüssel leitet jeder Knoten u seinen eigenen Master-Key ab. D.h. $K_u = f_{K_I}(u)$. Sind die Knoten dann im Einsatzgebiet allokiert, versucht jeder Knoten seine unmittelbaren Nachbarn zu finden. Eine kritische Annahme hierbei ist, dass die Zeit T_{min}, die ein Angreifer benötigt, um einen Knoten zu kompromittieren, größer ist als die Zeit T_{est} , die ein Knoten benötigt, um seine unmittelbaren Nachbarn zu finden und mit ihnen einen Schlüssel auszuhandeln. Der Knoten findet diese, indem er durch einen Broadcast eine HELLO-Nachricht verschickt, die seine ID (eindeutige Identitäsnummer) enthält und anschließend wartet, bis jeder Nachbarknoten v mit einer Bestätigungsnachricht, einem ACK antwortet, welcher die ID von v enthält. Die ACK-Nachricht von jedem Nachbar wird authentifiziert, indem der Master-Key von v benutzt wird. V schickt in seiner Nachricht seine ID unverschlüsselt und einen Message Authentication Code (MAC), parametrisiert mit dem Master-Key von v und den IDs von u und v. Dabei ist es egal, ob der MAC auf einer Hashfunktion mit einfacher XOR Operation oder auf einer Stromchiffre beruht, wichtig ist nur, dass er mit einem Geheimnis parametrisiert ist, das beide Knoten kennen (K_v). Der Ablauf sieht wie folgt aus (die Identitätsnummer der Knoten wird mit id bezeichnet) .

$u \rightarrow * : id_u$
$v \rightarrow u : id_v, MAC(K_v, id_u \| id_v)$

Knoten u kann jetzt durch den Erhalt von id_v v's Master-Key K_v berechnen, somit auf das Ergebnis des MAC's vergleichen und daraus v's Identität verifizieren. Anschließend kann Knoten u seinen paarweisen Schlüssel K_{uv} mit Knoten v als $K_{uv} = f_{K_v}(u)$ berechnen.

Nach einer gewissen Zeit löscht Knoten u den initialen Schlüssel K_I und alle Master-Keys K_v's seiner Nachbarn, die in der Erkennungsphase ermittelt wurden. Dies ist der kritische Zeitrahmen, in dem ein Angreifer noch die Möglichkeit hätte, das Netzwerk zu

kompromittieren. Ist die Löschung einmal erfolgt, bleibt nur der Master-Key K_u des Knoten selbst in dessen Speicher übrig. Dabei wird davon ausgegangen, dass es jedem Knoten möglich ist, die anderen Schlüssel rückstandslos zu löschen (das kann sich bei normalen Festplatten als schwierig erweisen, auf diese Problematik sei hier aber nicht weiter eingegangen). Das Verfahren funktioniert analog für jeden anderen Knoten, diese wiederum auch nur ihren Master-Key behalten. Ein Angreifer, der den Netzwerkverkehr mithören konnte, kann keine dieser Nachrichten entschlüsseln, da er den initialen Schlüssel K_I nicht besitzt.

Verteilung von Clusterschlüsseln — Die Verteilung von Clusterschlüsseln erfolgt auf die Verteilung von paarweisen Schlüssel und funktioniert nach dem selben Schema. Wenn ein Knoten u also mit allen seinen unmittelbaren Nachbarn $v_1, v_2, \ldots, v_m$ einen Clusterschlüssel aushandeln will, generiert er einen zufälligen Schlüssel K_u^c und verschlüsselt diesen anschließend mit den jeweiligen paarweisen Schlüssel seiner Nachbarn und verschickt ihn anschließend zu jedem seiner Nachbarn.

$$u \rightarrow v_i : (K_u^c)_{K_{uv_i}}$$

Knoten v_i entschlüsselt u's Cluster-Key K_u^c und speichert ihn. Anschließend schickt v_i nach dem gleichen Verfahren seinen Cluster-Key an u.

Verteilung von Gruppenschlüsseln — Eine Möglichkeit einen netzwerkweiten Gruppenschlüssel zu vereinbaren wäre, dass die Basisstation einen Gruppenschlüssel generiert und diesen jeweils einzeln mit jedem individuellen Schlüssel den sie mit jedem Knoten teilt, verschlüsselt. Da evtl. nicht alle Knoten in Kommunikationisreichweite sind, müssten andere Knoten sehr viele Nachrichten weiterleiten. Außerdem wäre der Berechnungsaufwand für die Basisstation enorm, hinzu kommt, dass falls die Basisstation die einzelenen indiviuellen Schlüssel für jeden Knoten nicht speichert, sondern wie oben beschrieben direkt aus ID und Master-Key berechnet, der Aufwand noch viel größer wäre.

Eine andere Möglicheit für die Basisstation eine Nachricht M an alle Knoten sicher zu übertragen, besteht in der Nutzung der Clusterschlüssel. Die Basisstation verschlüsselt die Nachricht mit ihrem Clusterschlüssel und verschickt einen Broadcast. Jeder benachbarte Knoten enschlüsselt die Nachricht, liest M, und verschlüsselt sie anschließend mit seinem eigenen Cluster-Schlüssel und verschickt ebenfalls einen Broadcast. Dies hat jedoch den Nachteil, dass jeder Zwischenknoten die Nachricht entschlüsseln und anschließend wieder verschlüsseln muss, was abträglich für die Effizienz des gesamten Netzwerks ist.

Ein Gruppenschlüssel ist riskant, da ein Verfahren benötigt wird, eine Aktualisierung (Re-Keying) des Gruppenschlüssels durchzuführen, sobald eine kompromittierter Knoten entdeckt wurde. Es sei angemerkt, dass auch bei einer Kompromitierung des Gruppenschlüssels Schaden nicht zwingend für sensible Daten entsteht, da Sensordaten zwischen den Knoten untereinander mit ihren paarweisen Schlüsseln codiert werden, und Daten, die bei der Basisstation weiter aggregiert werden sollen, mit dem individuellen Schlüssel oder dem Clustschlüssel verschickt werden können. Das das oben erwähnte Unicast Verfahren zu aufwendig ist, bleibt die Möglichkeit, nicht speziell jede einzelne Nachricht der Basisstation über Cluster-Keys zu codieren, sondern nur den Gruppenschlüssel über die Cluster-Keys zu verteilen. Der Gruppenschlüssel kann so auch in regelmäßigen Abständen geändert werden (Re-Keying). Da eine Änderung des Gruppenschlüssels nicht ständig erfolgt, hält sich der Kommunikations- und Berechnungsaufwand für die Sensorknoten in Grenzen. Ferner sei noch erwähnt, dass die Basisstation jede Nachricht authentifiziert verschickt, damit sich kein Angreifer als Basisstation ausgeben kann. Das Authentifikationsverfahren ist ein anderes wie das bereits beschriebene Authentifikationsverfahren der Knoten untereinander. Bei der

Basisstation kommt μTesla zur Broadcast Authentifikation zum Einsatz, auf dies hier nicht näher eingegangen wird.

3.4 Verfahren basierend auf verteilten Netzen

Der Anspruch an ein Verfahren, keine zentrale Instanz zur Koordination der Knoten zu verwenden, ist sicherlich hoch. Die Funktionsweise solche eines Verfahrens sei nachfolgend erläutert.

3.4.1 Entwurf von Eschenauer and Gligor - EG Schema

Einen gänzlich anderen Ansatz zur Schlüsselverteilung als bei LEAP wurde von L. Eschenauer und V.Gligor vorgestellt (nachfolgend EG Schema genannt) [EsV.02] . Dieses Verfahren kann in einem verteilten Netz eingesetzt werden, in dem die Knoten Ad-Hoc miteinander kommunizieren. Der Einsatz von Basisstationen und dergleichen wird nicht benötigt. Lediglich die Existenz von Kontroll-Knoten wird vorausgesetzt. Diese Knoten sind für das Re-Keying und teilweise für das Schlüsselmanagement zuständig. Das Verfahren wurde bereits 2002 vorgestellt, ist jedoch durchdacht und wurde bereits zahlreich modifiziert.

Das Verfahren an sich ist probabilistischer Natur, d.h. Knoten können nur zu einer gewissen Wahrscheinlichkeit einen Schlüssel untereinander aushandeln. Auch hier wird vor der Allokation der Sensorknoten eine Vorverteilung von Schlüsseln vorgenommen, jedoch auf andere Art und Weise. Zuerst wird ein großer Key-Pool P offline generiert, der alle möglichen gültigen Schlüssel für die Kommunikation enthält. Aus diesem Key-Pool wird anschließend eine Untermenge (nachfolgend Key-Ring genannt) von k Schlüsseln zufällig für jeden Knoten ausgewählt und auf diesem gespeichert. Die Anzahl der Schlüssel im Key-Ring wird dabei so gewählt, dass zu einer bestimmten Wahrscheinlichkeit zwei verschiedene Key-Rings mindestens einen gemeinsamens Schlüssel haben [ChPS03]. Durch die zufällige Auswahl des Key-Rings, ist es möglich, dass ein gemeinsamer Schlüssel zwischen zwei benachbarten Knoten nicht existiert. Ist dies der Fall, so könnten die Knoten dennoch einen Schlüssel aushandeln, sofern ein Pfad von Knoten a nach Knoten b über Zwischenknoten existiert, die jeweils untereinander einen gemeinsamen Schlüssel besitzen. Knoten a und Knoten b können dann über diesen Pfad einen direkte Verbindung herstellen, d.H. einen gemeinsamen Schlüssel aushandeln. Entscheidendes Kriterium bei diesem Ansatz ist die Schlüsselkonnektivität, die ausreichend hoch sein sollte. Im EG-Schema ist bereits ein Key-Ring von 250 Schlüsseln mit eine Key-Pool von 100.000 Schlüsseln ausreichend, um ein Netzwerk von 10.000 Sensorknoten mit sehr hoher Wahrscheinlichkeit vollständig zu verbinden. Die Einsparung an Speicherplatz ist enorm, was die Zahlen verdeutlichen.

Das Verfahren zur Schlüsselverteilung besteht generell aus drei Phasen. Diese sind:

Phase 1: Schlüssel-Vorverteilung — Diese Phase besteht aus wenigen Einzelschritten, die alle offline durchgeführt werden. Als erstes wird der Key-Pool mit den zugehörigen Schlüssel-ID's generiert, anschließend wird ein Key-Ring für jeden Knoten generiert welcher dann in Schritt drei auf jedem Knoten samt Schüssel-ID's gespeichert wird. Anschließend werden die Sensor-ID's der Knoten und Schlüssel-ID's des Key-Rings auf den Kontroll-Knoten gespeichert. Als letztes bekommt jeder Knoten einen eigenständigen Schlüssel mit den Kontroll-Knoten.

Phase 2: Gemeinsame Schlüsselermittlung — Diese Phase findet statt, sobald die Sesorknoten im Zielgebiet allokiert sind. Hier ist der einfachste Weg wieder ein Broadcast, der diesmal alle Schlüssel-ID's des Key-Rings des jeweiligen Knoten enthält. Ein Knoten u verschickt also fast analog zu LEAP

$$u \rightarrow * : id_{1..i \in Keyring}$$

In der Grundversion des EG-Schemas erfolgt hier im Gegensatz zu LEAP keine Authentifikation der Knoten untereinander. Es existieren jedoch Verbesserungen, die ähnlich zu LEAP eine Authentifikation über ein Challenge-Response Verfahren bzw. über einen MAC gewährleisten. Nachdem alle Knoten ihre Schlüssel-ID's an ihre Nachbarn verschickt haben, prüfen diese in ihrer ID-Tabelle, ob sie einen gemeinsamen Schlüssel (gleiche ID) teilen. Ist dies der Fall, so kann die Kommunikation fortan über diesen Schlüssel erfolgen. Teilen sich zwei Knoten mehrere Schlüssel, so wählen sie unter diesen einen zur Verschlüsselung aus.

Phase 3: Pfadschlüsselgenerierung — Diese Phase weist jedem Knoten einen Pfadschlüssel zu jedem anderen Knoten in Kommunikationsreichweite zu, falls diese Knoten nicht schon einen gemeinsamen Schlüssel teilen und ein wie bereits oben beschriebener Pfad über Zwischenknoten existiert, über die eine direkte Verbindung ausgehandelt werde könnte. Dies könnte nach einem vereinfachten Schema z.B. folgendermaßen geschehen: Die Menge der Nachbarknoten, mit denen ein Schlüssel geteilt wird, wird mit $Keyshare$ bezeichnet

$$u \rightarrow * : sensorid_{1..i \ni Keyshare}$$

Also ein Broadcast mit allen Sensorid's in der Nachbarschaft von u, mit denen u keinen gemeinsamen Schlüssel hat. Wenn ein Zwischenknoten w daraufhin feststellt, dass er einen gemeinsamen Schlüssel mit einem der Knoten v hat, den Knoten u nicht hat, kann er dies Knoten u durch die ensprechende Sensor-ID und ein ACK mitteilen. u könnte daraufhin einen Schlüssel K_{uv} für die Kommunikation mit v generieren und anschließend diesen Schlüssel mit dem gemeinsamen Schlüssel mit w codieren und dies an w schicken. Knoten w kann die Nachricht entschlüsseln und mit seinem gemeinsamen Schlüssel mit v codieren und dies an v Schicken. u und v sind dann im Besitz eines gemeinsamen Schlüssels.

$$u \rightarrow w : K_{uw}(K_{uv})$$
$$w \rightarrow v : K_{wv}(K_{uv})$$

Wiederrufen von Schlüsseln (Revocation) und Re-Keying

Das Widerrufen von Schlüsseln, sobald ein Knoten als kompromittiert festgestellt wurde, geschieht in LEAP über die Cluster- oder über den Gruppenschlüssel. Im EG-Schema nehmen diese Aufgabe die Kontroll-Knoten war. Der Kontroll-Knoten schickt eine Widerrufsnachricht per Broadcast an alle Knoten. Weiterhin generiert er einen Signatur-Schlüssel K_e , signiert die Schlüssel-ID's des kompromittierten Key-Rings damit und schickt an jeden Knoten einzeln per Unicast die signierte Liste der widerrufenen Schlüssel, codiert mit dem Schüssel des Kontrollknotens, den die einzelnen Sensorknoten bereits in Phase 1 der Schlüssvorverteilung erhalten haben. Jeder Sensorknoten entschlüsselt die Nachricht, verifiziert die Signatur und löscht die Schlüssel aus seinem Key-Ring (sofern vorhanden). Nachdem diese Schlüssel gelöscht wurden, kann es sein, das einige Verbindungen im Netz nicht mehr möglich sind, da die gelöschten Schlüssel nicht mehr zur Kommunikation verwendet werden können. Es folgt eine erneute Phase 2 (gemeinsame Schlüsselermittlung), in der sich die Knoten, sofern möglich, neu verbinden.

Re-Keying im EG-Schema ist denkbar einfach, es funktioniert analog wie ein Selbst-Widerruf von Schlüsseln eines Knoten, mit dem Unterschied, dass kein Broadcast von den

Kontrollknoten notwendig ist. Nachdem bestimmte Schlüssel abgelaufen sind, starten die betroffen Knoten Phase 2 und evtl. darauf Phase 3 einfach neu.

3.4.2 Verbesserte Derivate des EG-Schemas

Das sehr erfolgreiche EG-Schema kann an bestimmten Stellen noch verbessert werden. Zwei wichtige Verbesserungen des EG-Schemas seien nachfolgend erläutert.

Das q-Verbund Schema

Im Basismodell des EG-Schemas mussten zwei benachbarte Knoten nur einen Schlüssel finden um eine Verbindung zu etablieren. Das q-Verbund Schema modifiziert das EG-Schema dahingehend, dass nun mehrere gemeinsame Schlüssel zwischen zwei Knoten nötig sind, um eine Verbindung zu etablieren. In dem die Anzahl der sich überschneidenden Schlüsselmenge erhöht wird, erhöht sich auch die Widerstandsfähigkeit des Netzwerks gegen Knotenkompromittierung. Diese Schema hat allerdings den Nachteil, das die Größe des Key-Pools P entweder verringert werden muss oder die Anzahl der Schlüssel in den Key-Rings der Knoten erhöht werden muss. Ersteres hat zur Folge, dass es einem Angreifer dann leichter fallen wird, eine größere Untermenge von P zu erhalten, während er gleichzeitig weniger Knoten im Netz kompromittieren müsste. Zweiteres hat einen erhöhten Speicherverbrauch bei den Sensorknoten zur Folge. Die Ermittlung der optimalen Größe des Key-Pools P und der Key-Rings ist nicht trivial und wird von Haowen Chan, Adrian Perrig und Dawn Song [ChPS03] näher erläutert. Nachdem also in Phase 2 jeder Knoten die Nachbarknoten bestimmt hat, mit denen er mindestens q Schlüssel teilt, kann ein neuer Schlüssel aus dieser gemeinsamen Schnittmenge errechnet werden. Dies kann z.b. über eine Hash-Funktion über alle gemeinsamen Schüssel erfolgen.

$$K = hash(k_1 \parallel k_2 \parallel \ldots \parallel k_q)$$

Bei Knoten, mit denen weniger als q Schlüssel geteilt werden, wird keine Verbindung aufgebaut. Diese Knoten werden später dann über Phase 3 miteinander Verbunden. Alternativ kann nicht nur ein Hash von der Mindestanzahl q von Schlüsseln berechnet werden, sondern über die Anzahl an Schlüssel, die tatsächlich zwischen zwei Knoten geteilt werden (wobei diese Anzahl natürlich größer als q sein muss)

Eine weitere Methode die in diesem Schema vorgeschlagen wird, um die Sicherheit zu erhöhen, ist das in Phase 2 der Broadcast der Key-Id's zusätzlich mit einem Puzzle versehen wird, d.h. einer Rechenaufgabe, die nur unter zusätzlichen Informationen lösbar ist (diese könnten auf den Knoten vorverteilt sein). Verbindungen könnten dann nur von Knoten akzeptiert werden, die mit der richtigen Lösung auf das Puzzle antworten und sich somit authentifizieren. Bei diesem Ansatz würde Rechenkapazität und Kommunikationseffizienz zugunsten von Sicherheit, sprich zusätzlicher Authentifikation, geopfert werden.

Schlüsselverstärkung über mehrere Pfade - Multi Path Key Reinforcement

Dieser Ansatz modifiziert das EG-Schema dahingehend, dass Schlüssel zwischen den Knoten jetzt über mehrere Pfade etabliert werden. Die Grundidee für diese Modifikation stammt ursprünglich von R. Anderson and A. Perrig [AnPe01]. Es wird versucht, folgendes Problem zu lösen (nachfolgend wird wieder vom EG-Basisschema ausgegegangen): Nachdem alle Knoten ihren gemeinsamen Schlüssel mit ihren Nachbarn ermittelt haben, besteht mit recht hoher Wahrscheinlichkeit die Möglichkeit, dass dieser Schlüssel auch in dem Key-Ring zahlreicher

anderer Knoten im Netz ist. Ist demnach einer dieser Knoten kompromittiert, so ist die Sicherheit der ursprünglichen Verbindung gefährdet. Zur Verdeutlichung nehmen wir einmal an, Knoten u hat mit Knoten v einen gemeinsamen Schlüssel, der für die Kommunikation verwendet wird. Dieser Schlüssel könnte jetzt auch im Key Ring eines dritten Knoten w enthalten sein, obwohl er dort für w keine Verwendung findet. Wird w nun kompromittiert, so ist die Sicherheit zwischen u und v insofern gefährdet, dass ein Angreifer zumindest die Möglichkeit hat, den Schlüssel K_{uv} durch einfaches sukzessives Ausprobieren (Brute-Force) aller Schlüssel im Key-Ring von w auf die Knoten im Netzwerk, zu finden imstande ist.

Das Verfahren geht von der Annahme aus, dass genügend Routing-Informationen zur Verfügung stehen, sodass ein Knoten u alle verschiedenen Wege über andere Knoten zu v kennt. Ein Pfad wird nur verwendet, wenn folgende Bedingung gilt: Die Knoten $u, w_1, w_2, \ldots, w_i, v$ sind im Pfad durch paarweise gemeinsame Schlüssel verbunden. D.h. die Tupel

$$\{(u, w_1), (w_1, w_2), \ldots, (w_{i-1}, w_i), (w_i, v)\}$$

haben jeweils für sich einen gemeinsamen Schlüssel. Sei j nun die Anzahl der Pfade, die die obige Bedingung nicht erfüllen. Knoten u generiert dann j zufällige Schlüssel $v_1, \ldots, v_j$, wobei jeder Schlüssel v die gleiche Länge hat als ein normaler Schlüssel. u schickt diese Schlüssel jetzt jeweils entlang einem unterschiedlichen Pfad zu v, der die obige Bedingung erfüllt. Wenn v alle j Schlüssel erhalten hat, können die beiden Knoten u und v ihren gemeinsamen neuen Schlüssel k' z.B. durch simple XOR Operationen berechnen:

$$k' = k \oplus v_1 \oplus v_2 \oplus \ldots \oplus v_j$$

mit k als initialem gemeinsamen Schlüssel. Die Sicherheit des Schlüssels k', wird jetzt von allen verschiedenen Pfaden geschützt, über die $v_1 \ldots v_i$ geschickt wurden. Da Bitweise XOR Operationen nicht berechnungsintensiv sind, hält sich die Prozessorbelastung für die Sensorknoten in Grenzen. Durch die vielen unterschiedlichen Pfade entsteht eine höhere Widerstandsfähigkeit gegen Knotenkompromittierung, jedoch wird dies durch einen nicht zu vernachlässigenden Kommunikationsoverhead erkauft.

3.5 Diskussion

Vor- und Nachteile der vorgestellten Verfahren werden nachfolgend diskutiert und evaluiert. Zuerst werden die EG-Schema Modifikationen untereinander verglichen, anschließend das LEAP Verfahren mit dem EG Schema.

3.5.1 EG-Schema Modifikationen

Es stellt sich also die Frage, welche der beiden Modifikationen des EG-Schemas eine höhere Sicherheit bietet. Nachfolgend eine Evaluierung von Haowen Chan et al., die einen Einblick gibt, wie sich die Widerstandsfähigkeit gegen Knotenkompromittierung mit den jeweiligen Modifikationen erhöht. In der Auswertung wurden auch beide Modifikationen miteinander kombiniert. Die Grafik beruht auf einer Key-Ring Größe von 200 Schlüsseln und einer Schlüsselkonnektivität von 33 %.

Es fällt auf, dass die Schlüsselverstärkung über mehrere Pfade einen wesentlich höhere Sicherheit gegen Knotenkompromittierung bietet als das q-Verbund Schema (mit $q > 2$). Sogar beide Modifikationen miteinander kombiniert, haben immer noch eine höhere Anfälligkeit als die Schlüsselverstärkung über mehrere Pfade für sich genommen. Dies liegt daran, dass das Prinzip hinter beiden Methoden das gleiche ist, nur die Realisierungen verschieden. Ziel bei beiden Modifikationen ist es, es einem Angreifer schwerer zu machen, die Kommunikation

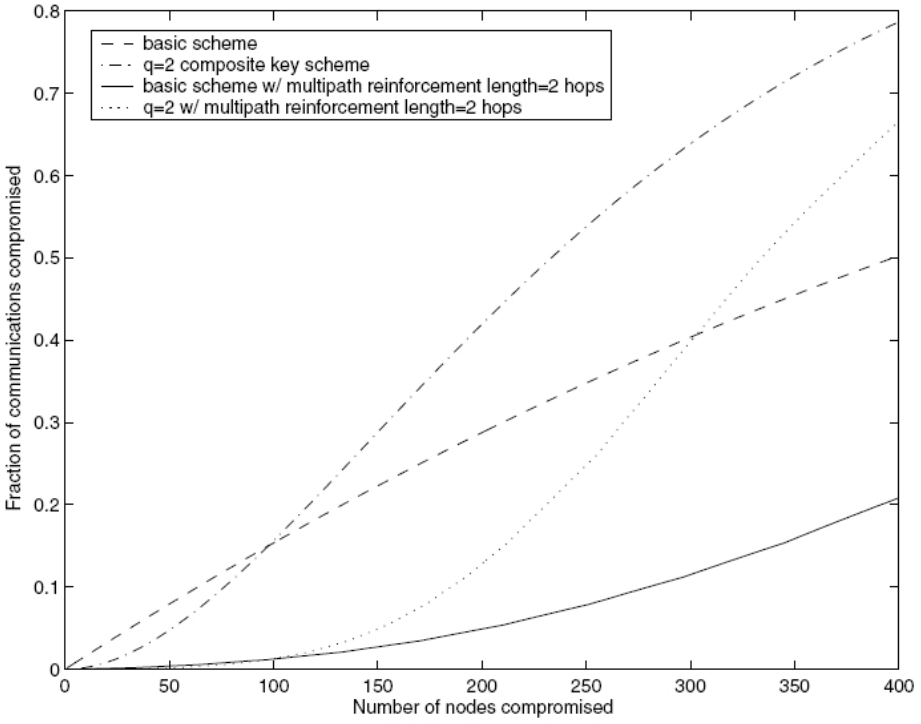

Abbildung 1: Widerstandsfähigkeit gegen Knotenkompromittierung

zwischen zwei Knoten bzw. im Netzwek zu entschlüsseln, indem er, um dies zu tun mehrere relevante Schlüssel besitzen muss. Da, wie bereits erwähnt, sich beim q-Verbund Schema der Key-Pool verringert - als Alternative die Erhöhung des Key-Rings der Knoten ist wegen der limitierten Speicherkapazität nur begrenzt möglich - und sich bei der Schüsselverstärkung über mehrere Pfade der Kommunikationsoverhead erhöht, hat dies zur Folge, dass bei einer Kombination beider Modifikationen hauptsächlich ihre Schwachpunkte miteinander kombiniert werden. Die geringer Key-Pool Größe macht die Schlüsselverstärkung über mehrere Pfade überproportional schwächer.

3.5.2 LEAP und das EG-Schema

Ein Vergleich des LEAP Verfahrens mit dem des EG-Schemas ist sicherlich problematisch, da sie auf unterschiedlichen Netzwerkmodellen beruhen und unterschiedliche Grundannahmen voraussetzen. Dennoch soll hier versucht werden, zumindest einen groben Überblick zu erhalten. Die Verfahren werden anhand der in Kapitel 2 erläuterten Sicherheitsziele und Anforderungskriterien beurteilt.

Vertraulichkeit — Vertraulichkeit, eines der wichtigsten Sicherheitsziele, wird zweifelsohne von beiden Verfahren erreicht. Bei beiden Verfahren beginnt die Verschlüsselung augenblicklich, nachdem die Sensorknoten ihre unmittelbaren Nachbarn entdeckt haben und sich auf einen gemeinsamen Schlüssel geeinigt haben. Welches symmetrische Verschlüsselungsverfahren dafür eingesetzt wird und wie lang ein Schlüssel ist, sei dahingestellt und ist weitgehend von der Rechenkapazität der Sensorknoten abhängig.

Authentizität — Authentifikation wird bei LEAP bereits in der Phase durchgeführt, in der Sensorknoten ihre paarweisen gemeinsamen Schlüssel aushandeln. Authentifiziert wird

dabei über einen MAC und einem vorverteilten Geheimnis (dem Master Key des antwortenden Knoten, der sich aus dem initialem Schlüssel K_I ableitet). Diese Phase entspricht im EG-Schema Phase 2, in der Knoten ebenfalls ihre gemeinsamen Schlüssel aushandeln. In der Grundversion des EG-Schemas erfolgt in dieser Phase keine Authentifikation der Knoten untereinander, es wird von L. Eschenauer und V.Gligor jedoch vorgeschlagen, dies über ein einfaches Challenge-Response Verfahren zu lösen, in dem jeder Knoten einen Wert jeweils mit unterschiedlichen Schlüsseln aus seinem Key-Ring codiert, und diese verschickt. Antwortet ein Knoten mit der richtigen Entschlüsselung, hat er sich damit authentifiziert. Dieses Verfahren hat jedoch den Nachteil, dass sich die Schlüsselkonnektivität verringern würde. Die Verringerung wäre hierbei abhängig von Anzahl der verschickten unterschiedlich verschlüsselten Nachrichten.

Integrität — Integrität kommt mit den bereits genannten Zielen Vertraulichkeit und Authentizität einher. So kann die Integrität einer Nachricht als bewiesen angesehen werden, sofern die Nachricht entschlüsselt werden konnte. Dies setzt die Annahme voraus, dass diese Nachricht nicht von einem Dritten entschlüsselt, verändert und anschließend wieder verschlüsselt verschickt wurde. Desweiteren kann Integrität gewährleistet werden, wenn Nachrichten unverschlüsselt übertragen werden und zusätzlich ein MAC mit vorverteiltem Geheimnis über diese Nachricht angehängt wird. Der Empfänger kann von der unverschlüsselten Nachricht ebenfalls einen MAC-Wert herleiten und auf den in der Nachricht mitverschickten vergleichen. Sowohl LEAP als auch das EG-Schema können Integrität gewährleisten. LEAP zusätzlich über einen MAC. Das EG-Schema in der Grundversion lediglich über die Verschlüsselung der Nachrichten.

Autorisation, Verbindlichkeit und Verfügbarkeit — Autorisation impliziert, dass eine Klassifikation von Knoten existieren würde, bei der manche Knoten mehr Rechte als andere Knoten haben. Dies ist bei den vorliegenden Verfahren nicht der Fall, sieht man bei LEAP einmal von der Basisstation und im EG-Schema von den Kontrollknoten ab. Diese jedoch sind dadurch autorisiert, dass sie sich authentifizieren. Eine feinere Abstufung bzw. eine Rechtevergabe für normale Sensorknoten war in Sensornetzen allgemein, wie auch bei LEAP und dem EG-Schema bisher kein Designziel. Es sei jedoch angemerkt, dass sich dies leicht durch Access-Control-Lists zum Nachteil des Speicherverbrauchs realisieren ließe. Weiterhin wird Verbindlichkeit im Anwendungsfall von Sensornetzen abgelehnt, das sie nur bei Vertragsverhandlungen und derleichen von Bedeutung ist. Verfügbarkeit und Überlebensfähigkeit der Sensorknoten ist wichtig. Diese Ziele hängen jedoch primär von Bauart und Akkukapazität der Sensorknoten ab, auf die das Schlüsselverteilungsverfahren keinen Einfluss hat. Sekundär hängt die Laufzeit eines Knotens jedoch von seiner Kommunikationstätigkeit ab, worauf das Schlüsselverteilungsverfahren einen Einfluss hat. Die Unterschiede werden nachfolgend noch erläutert.

Qualität der Sicherheitsdienstgüte — Hier ist LEAP klar im Vorteil, da es das Sicherheitsniveau über mehrere verschiedene Schlüsselarten anpassen kann. Im EG-Schema hingegen werden alle Nachrichten als gleich wichtig erachtet.

Widerstandsfähigkeit gegenüber Schlüsselkompromittierung — Dies ist schwer zu beurteilen. Man kann jedoch festhalten, dass die Wiederstandsfähigkeit im EG-Schema durch die Schlüsselverstärkung über mehrere Pfade signifikant erhöht werden kann. LEAP hat eine sehr hohe Widerstandsfähigkeit gegen Knotenkompromitierung, sofern nicht noch während der gemeinsamen Schlüsselaushandlungsphase Knoten kompromitiert werden. Hauptnachteil

von LEAP aber ist sicherlich, dass es von der Annahme ausgeht, die Basisstation wäre kompromittierungssicher.

Konnektivität der Schlüssel — Ist bei LEAP immer gegeben, da es kein probabislistisches Verfahren ist. Im EG-Schema kann eine volle Konnektivität des Netztes ebenfalls erreicht werden, auch wenn die Schlüsselkonnektivität wesentlich niederiger als eins ist. Bei probabilistischen Verfahren kommt jedoch erschwerend hinzu, dass die optimale Key-Pool Größe und die Größe des Key-Rings in Abhängigkeit der Größe des Sensornetzwerks bestimmt werden muss.

Skalierbarkeit — Auch die Sendereichweite einer Basisstation ist begrenzt. Ist bei LEAP das Sensornetz sehr groß, müssten mehrere Basisstationen zum Einsatz kommen. Der Koordinationsaufwand würde bei vorher nicht bekannter Netzwerktopologie erheblich Ansteigen, da je nach Basisstation unterschiedliche Schlüssel auf den Sensorknoten vorverteilt werden müssten. Dies ist im EG-Schema nicht der Fall, womit es einen erheblichen Skalierbarkeitsvorteil gegenüber LEAP hat.

Effizienz —

- Berechnunsaufwand:
  Im EG-Schema entsteht lediglich ein Suchaufwand, da Schlüssel-IDs im Key-Ring gesucht werden müssen. Bei LEAP kommt die Berechnung eines MAC zusätzlich zum Suchaufwand dazu. Bei der EG-Modifikation mit Schlüsselverstärkung über mehrere Pfade würden zum Suchaufwand noch XOR Operationen hinzukommen. Da ein MAC meist mehrere XOR Operationen enthält, bleibt festzuhalten, dass die Berechnunskomplexität bei LEAP höher ausfällt.
- Speicheraufwand:
  Bei LEAP müssen nur wenige Schlüssel gespeichert werden. Der Individuelle Schlüssel mit der Basisstation. Die ausgehandelten Schlüssel mit dem Nachbarn plus anfänglich der initiale Schlüssel. Der eigene Clusterschlüssel und die Clusterschlüssel der Nachbarn und der durch die Clusterschlüssel ausgehandelte Gruppenschlüssel. Im EG-Schema muss in jedem Fall der Key-Ring gespeichert werden, die Schlüssel mit den Kontroll-Knoten und evtl. einige neue Schlüssel, die in Phase 3 des EG-Schemas erzeugt wurden. Bei welchem Verfahren mehr Speicheraufwand entsteht, ist von der Größe des Key-Rings im EG-Schema abhängig.
- Kommunikationsaufwand:
  LEAP hat durch die Vielzahl von verschiedenen Schlüsseln, die ausgehandelt werden müssen, schon aufgrund der Architektur einen höheren Kommunikationsbedarf zwischen den Knoten. Da im EG-Basisschema nur eine gemeinsamer Schlüssel ausgehandelt werden muss, ist der Kommunikationsaufwand im EG-Schema geringer. Werden hingegen über mehrere verschiedene Pfade (Schlüsselverstärkung) Schlüssel ausgehandelt, erhöht sich auch hier der Kommunikationsaufwand in Abhängigkeit der Anzahl der unterschiedlichen Pfade, über die Schlüssel ausgehandelt werden.

4 Fazit

Es scheint klar zu sein, dass die Geeignetheit eines Verfahrens von dem vorgesehenen Einsatzgebiet abhängt. Ist die Gefahr eines physischen Angriffs groß, diesem auch eine Basisstation ausgesetzt sein könnte, so ist ein dezentrales Verfahren wie das EG-Schema

sicherlich vorzuziehen. Auch wenn die Sensorknoten über eine große Distanz verteilt sind und ein großes Gebiet abdecken, ist LEAP sicherlich die schlechtere Wahl. Ist die physische Angriffgefahr jedoch gering und die erhöhte Gefahr eines Man-in-the-Middle Angriffs gegeben, könnte LEAP mit seiner bereits integrieten Authentifikation die bessere Alternative darstellen. Ferner hängt die Wahl des Schlüsselverteilungsmechanismus natürlich auch von der Leistungsfähigkeit der Sensorknoten ab.

Die beiden vorgestellten Verfahren sind nur ein kleiner, aber vielversprechender Teil der Verfahren, die gegenwärtig existieren. So wurde, um ein Beispiel zu nennen, mit HARPS (Hashed Random Preloaded Subsets) eine neue Variante vorgestellt die verschiedene Schemata miteinander kombiniert, ebenfalls auf einem verteilten Netzwerk beruht und ebenso wie das EG-Schema auf Vorverteilung von Schlüsseln basiert [RaMe05]. Da Sensortechnologie noch ein Recht junges Forschungsgebiet darstellt, ist sie einem raschen Wandel unterworfen. Es bleibt zu vermuten, dass sich kommende Verfahren und der Standard, der sich auf diesem Gebiet etablieren wird, sich primär an der Leistungsfähigkeit der Sensorknoten orientieren wird, um somit den besten Kompromiss zwischen Effizienz und Sicherheit zu finden.

Literatur

[AnPe01] R. Anderson und A. Perrig. Key infection: Smart trust for smart dust. *Unpublished Manuscript*, 2001.

[BuDe94] M. Burmester und Y. Desmedt. A secure and efficient conference key distribution system. *Eurocrypt*, 1994.

[CaYe05] Seyit A. Camtepe und Bulent Yener. Key Distribution Mechanisms for Wireless Sensor Networks: a Survey. *Report TR-05-07, Rensselaer Polytechnic Institute*, März 2005, S. 2.

[ChPS03] H. Chan, A. Perrig und D. Song. Random Key Predistribution Schemes for Sensor Networks. *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003, S. 1.

[EsV.02] L. Eschenauer und V.Gligor. A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM Conference on Computer and Communication Security*, 2002, S. 41–47.

[NAMR+08] A. Naureen, A. Akram, T. Maqsood, R. Riaz, Kim Ki-Hyung und H.F. Ahmed. Performance and Security Assessment of a PKC Based Key Management Scheme for Hierarchical Sensor Networks. *Vehicular Technology Conference, VTC Spring 2008. IEEE*, Mai 2008, S. 165.

[PeSW04] Adrian Perrig, John Stankovic und David Wagner. Security in Wireless Sensornetworks. *Communications of the ACM*, Juni 2004, S. 1.

[RaMe05] Mahalingam Ramkumar und Nasir Memon. Harps - An Efficient Key Pre-distribution Scheme for Ad Hoc Network Security. *IEEE Journal on selected areas in communications, vol. 23, no. 3*, 2005.

[Resc99] E. Rescorla. Diffie-Hellman Key Agreement Method. RFC 2631, Juni 1999.

[StTW00] M. Steiner, G. Tsudik und M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 2000.

[Xbow08] Xbow. Product Feature Reference Chart 2008. *http://www.xbow.com*, 2008.

[ZhSJ04] Sencun Zhu, Sanjeev Setia und Sushil Jajodia. LEAP: Efficient Security Mechanisms for LargeScale Distributed Sensor Networks. *Report of ACM Conference*, August 2004, S. 2.

Abbildungsverzeichnis

Sicherheit in Peer-to-Peer-Netzen

Kai Richter

Seit einigen Jahren benutzen viele Anwender File-Sharing Peer-to-Peer-Netzwerke zum Austausch von Dateien. Größtenteils kümmern sie sich nicht um die steigende Bedeutung der Sicherheit im, für sie, transparenten Peer-to-Peer-System. Insbesondere ist die eigene Privatsphäre des Benutzers und Teile des kompletten Netzwerks betroffen. Zum Beispiel könnte ein Angreifer den Routing-Mechanismus eines Netzwerks beeinflussen, einem Netzteilnehmer falsche Ergebnisse einer Stichwortsuche zurückliefern oder Teilnehmerdaten für andere Benutzer verweigern. Ein Angreifer benötigt zur Durchführung der präsentierten Beispiele lediglich ein PC-System mit einer Bandbreite von 100 Mbps ins Internet. Anhand von zwei verbreiteten Angriffsmethoden mit unterschiedlichen Varianten wird gezeigt, welche Auswirkungen auf die Anwender und das Netzwerk entstehen.

1 Einleitung

Peer-to-Peer Systeme werden heutzutage von vielen Benutzern eingesetzt. Sei es zum Austausch von Dateien (**File-Sharing-Systeme**), zur Unterstützung der Zusammenarbeit innerhalb einer Arbeitsgruppe (Groupware-Systeme), zur anonymen Veröffentlichung von Dokumenten (Freenet) oder zum sicheren Speichern von Dateien (Secure-Distributed-Storage-Systeme). Bei solchen Systemen steht die gemeinsame Nutzung von Ressourcen und die Interaktion von Endsystemen (Peers) im Vordergrund. Das Thema der Sicherheit ist hier nicht vorrangig.

Das Besondere, im Gegensatz zu einer typischen Client-Server-Architektur[1], ist die Selbstorganisation des Netzes. Also können die Dienste der Peer-to-Peer Systeme ohne eine zentrale Kontrolle genutzt werden.

Gegenstand folgender Kapitel sind in der Praxis aktive File-Sharing Peer-to-Peer-Systeme, die auf dem Kademlia-Netzwerkprotokoll beruhen. Des Weiteren finden spezielle Angriffsmethoden gegen dieses Protokoll Anwendung.

1.1 Gliederung

Im zweiten Kapitel wird die Idee des Kademlia-Protokolls vorgestellt. Danach die Funktionsweise von Kademlia beschrieben, auf die später im dritten Kapitel zurückgegriffen wird. Darüber hinaus werden Anforderungen beim Design eines solchen Peer-to-Peer-Netzwerk betrachtet.

Das dritte Kapitel erklärt im Detail zwei sehr bekannte Angriffe auf Kademlia-Netzwerke und stellt einige Messergebnisse aus der Praxis dar.

Abschließend gibt das letzte Kapitel ein Fazit über die Sicherheit von File-Sharing Peer-to-Peer-Netzwerken und endet mit einer Schlussbemerkung in Blick auf die Zukunft.

[1] Bei einer Client-Server-Architektur stellt ein zentraler Server Dienste (Dateiablage, Email-Postfächer, etc.) bereit, auf welche viele Clienten über ein Netzwerk zugreifen können.

2 Kademlia basierende Peer-to-Peer-Systeme

2.1 Kademlia

Kademlia [MaMa02] ist ein Protokoll für Peer-to-Peer-Netze. Es hat die folgenden Ziele:

- Die Art und Struktur eines Peer-to-Peer-Netzwerks aufzubauen (siehe Abschnitt 2.1.1)
- Einen Routingmechanismus zur Wegewahl eines Datenstroms bereitzustellen (Details in Abschnitt 2.1.2)
- Operationen zum Speichern und Abrufen von Daten anzubieten (weiteres in Abschnitt 2.1.3)

File-Sharing-Applikationen, die dieses Kademlia-Protokoll umsetzen, sind unter anderem die Anwendungen eMule[2], aMule[3] und Azureus[4].

2.1.1 Art und Aufbau eines Peer-to-Peer-Netzwerks

Kademlia bildet ein selbstorganisierendes und strukturiertes Peer-to-Peer-Netz. Dazu wird das Kademlia-Netz als sogenanntes Overlay-Netzwerk oberhalb der Internet-Infrastruktur positioniert. Die Teilnehmer im Overlay-Netzwerk kommunizieren[5] über eine virtuelle oder logische Verbindung, welche über einen anderen Pfad abläuft als die unterliegende Topologie vorgibt. Im Overlay-Netzwerk stellt Kademlia einen eigenen Routingmechanismus (siehe Abschnitt 2.1.2) bereit. Außerdem haben die Teilnehmer im Kademlia-Netz eine eigene Adressierung. Teilnehmer (auch Clienten oder Knoten genannt) adressieren sich über eine eindeutige Kademlia-Identität (**Kad-Id**). Somit haben sie einen festen Platz im Kademlia-Netzwerk. Die genannte Identität besteht aus einer 160bit langen Nummer und wird von Kademlia innerhalb eines binären Präfix-Baumes verwaltet. Das gesamte Kademlia-Netz wird über einen binären Präfix-Baum mit allen Teilnehmerknoten repräsentiert. Abbildung 1 veranschaulicht die Position einer Kad-Id mit Präfix 001 in einem binären Präfix-Baum. Die eingezeichneten Kreise stellen Teilbäume des Präfix-Baums dar. Kademlia stellt sicher, dass die Kad-Id mindestens ein Knoten aus jedem seiner Teilbäume kennt. Genau diese Kontaktinformationen sind für späteres Routing und zum Speichern und Abrufen von Daten aus dem Kademlia-Netz wichtig (vgl. [MaMa02]).

2.1.2 Routingmechanismus

Innerhalb des Kademlia-Protokolls werden Nachrichten zwischen Teilnehmerknoten ausgetauscht. Welchen Pfad diese Nachrichten nehmen, entscheidet ein sogenannter Routingmechanismus.

Der Routingmechanismus im Kademlia-Netzwerk gewährleistet, dass die vorgenannten Nachrichten über andere Teilnehmerknoten zum Empfänger weitergeleitet werden. Dazu nutzt der Routingmechanismus die XOR-Metrik über die Kad-Ids zur Unterscheidung, ob ein Pfad über Teilnehmer a oder b kürzer ist. Um vorbezeichnete Unterscheidung berechnen zu können, braucht der Mechanismus Kontaktinformationen über andere Netzteilnehmer.

Die **XOR-Metrik** als Distanzfunktion ist für zwei Teilnehmer folgendermaßen definiert:

[2] http://www.emule-project.net
[3] http://www.amule.org
[4] http://azureus.sourceforge.net
[5] Zur Kommunikation verwendet Kademlia ausschließlich das UDP-Protokoll.

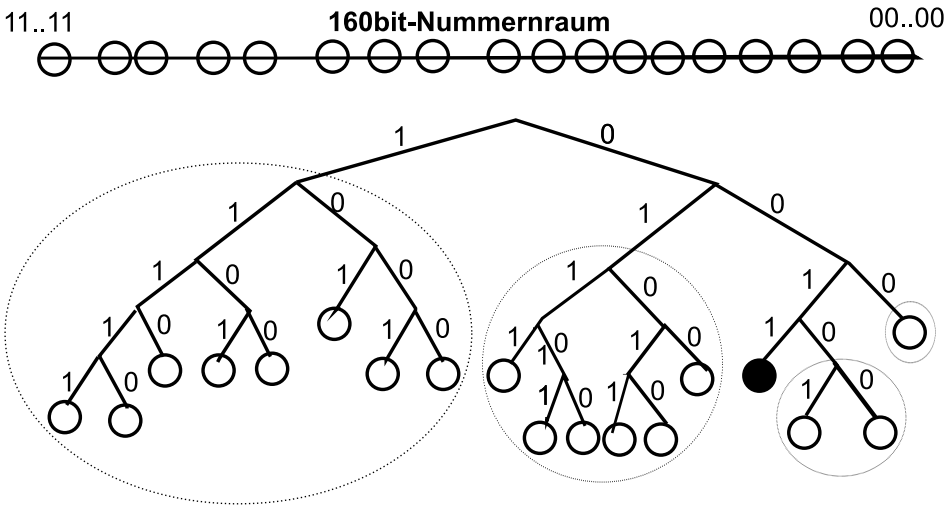

Abbildung 1: Binärer Präfix-Baum in Kademlia. Der angedeutete 160bit-Nummernraum zeigt die von 0 bis $2^{160} - 1$ möglichen Teilnehmerknoten. Einige Teilnehmer sind im Präfix-Baum eingetragen. Gezeigt wird hier die Kad-Id 0011... (gefüllter Kreis) und ihre dazugehörigen Teilbäume mit Präfix 1, 01, 000, 0010.

$$\text{Distanz(Kad-Id Teilnehmer a, Kad-Id Teilnehmer b)} = \text{Kad-}Id_a \text{ XOR Kad-}Id_b$$

Hierzu ein Beispiel: Es beteiligen sich vier Teilnehmer mit Kad-$Id_1 = 0101$, Kad-$Id_2 = 0011$, Kad-$Id_3 = 1110$ und Kad-$Id_4 = 1011$ im Kademlia-Netz. Eine Nachricht, adressiert mit Kad-Id_4, von Teilnehmer 1 soll Teilnehmer 4 erreichen. Der Teilnehmer 1 hat als Kontakte die Kad-Ids von Teilnehmer 2 und 3. Nur die Teilnehmer 2 und 3 haben Kenntnis über Teilnehmer 4. Der Routingmechanismus berechnet, initiiert von Teilnehmer 1, die folgenden Werte über die XOR-Operation:

$$Distanz(Id_4, Id_2) = Distanz(1011, 0011) = 1011\ XOR\ 0011 = 1011 = 11_{10}$$

$$Distanz(Id_4, Id_3) = Distanz(1011, 1110) = 1011\ XOR\ 1110 = 1111 = 15_{10}$$

Die kleinste Dezimalzahl, hier 11, wertet der Mechanismus als bessere Distanz. Daraufhin fragt Teilnehmer 1 bei Teilnehmer 2 nach Kontakten, die laut XOR-Metrik die bessere Distanz zu Kad-Id_4 liefern. Teilnehmer 2 kennt direkt Kad-Id_4. Somit hat Teilnehmer 1 den Pfad zu Teilnehmer 4 gefunden und kann obige Nachricht an Teilnehmer 4 senden.

In obigem Beispiel geht der Routingmechanismus von bereits bekannten Kad-Ids aus. Allerdings kann der Startknoten aus Speicherplatz- und Performanzgründen nicht die komplette Sichtweise (wie in Abschnitt 2.1.1) auf alle Kad-Ids liefern. Deswegen wird wie folgt eine **Routing-Tabelle** pro Teilnehmerknoten verwaltet:

Innerhalb einer Routing-Tabelle werden Informationen (IP-Adressen, UDP-Ports, Kad-Ids) über andere Teilnehmerknoten gespeichert. Um den Nummernraum der Kad-Ids von 0..2^{160}-1 zu verwalten, muss dieser Raum für die Routing-Tabelle in Abschnitte von je 2^i bis 2^{i+1} mit $0 \leq i < 160$ aufgeteilt werden. Dadurch hat eine Routing-Tabelle 160 Listen. Eine solche Liste hat Platz für k Teilnehmerkontakte und wird als **k-Bucket** bezeichnet[6]. Jedes i-te k-Bucket enthält nur die Teilnehmer, die um 2^i bis 2^{i+1} von der Kad-Id der Routing-Tabelle entfernt sind. Außerdem ist jedes k-Bucket nach dem letzten Kontaktzeitpunkt sortiert. Durch diesen Aufbau strukturiert sich die Routing-Tabelle als ein binärer Präfix-Baum, dessen Blätter die

[6] üblicherweise wird k=20 gewählt

k-Buckets sind. Die Kreise in der bereits gezeigten Abbildung 1 entsprechen den k-Buckets. Alle k-Buckets zusammen decken den gesamten 160bit-Nummernraum ab.

Die Buckets werden nach dem folgenden Algorithmus dynamisch gefüllt:

1. Zunächst besteht der binäre Präfix-Baum der Routing-Tabelle aus einem k-Bucket, das der eigenen Kad-Id entspricht. Das einzelne k-Bucket repräsentiert den gesamten 160bit-Nummernraum.
2. Wenn ein neuer Teilnehmerknoten bekannt wird, versucht ihn der Algorithmus in ein passendes k-Bucket wie folgt einzufügen:
   - Ist der entsprechende k-Bucket nicht voll, wird der neue Kontakt in der Liste gespeichert.
   - Ist der k-Bucket voll und die eigene Kad-Id dort enthalten wird der k-Bucket zweigeteilt und die alten Einträge auf die neuen k-Buckets verteilt. Der Einfügealgorithmus wird von Schritt 2 mit dem neuen Kontakt wiederholt.

Die beschriebene XOR-Metrik und die Routing-Tabellen sind für den Routingmechanismus unerlässlich. In Abbildung 2 wird ein Routingablauf eines Teilnehmers zu einem anderen Knoten dargestellt (**Knotensuche**). In diesem Beispiel findet die Kad-Id_a mit dem Präfix 0011 die Kad-Id_b mit Präfix 1110. Die Lokation funktioniert dadurch, dass Kad-Id_a aus seinem k-Bucket mit Präfix 1 die α nächsten Knoten parallel anfragt (Routen-Anfrage)[7]. Der Mechanismus wählt für Kad-Id_a genau diesen k-Bucket mit Präfix 1, weil dessen Inhalte laut XOR-Metrik die geringsten metrischen Abstände zur Kad-Id_b enthalten. Als Routen-Antwort erhält der Knoten mit Kad-Id_a weitere Kontakte von den α angefragten Knoten zurück. Die neuen Kontakte liegen laut XOR-Metrik näher am Ziel von Kad-Id_b als die alten Kontakte. Kad-Id_a speichert die neuen Kontakte in ihrer Routing-Tabelle ab. Dieses Verfahren wiederholt Kad-Id_a und endet erst bis in einem Durchlauf keine weitere Annäherung an Kad-Id_b mehr erreicht wird.

2.1.3 Speichern und Abrufen von Daten

In Kademlia können die Teilnehmer Informationen in Form von **Schlüssel-Wert-Paaren** ablegen und wieder abrufen. Eine Hashfunktion berechnet einen eindeutigen 160bit-Hashwert (Schlüssel) aus dem Informationsinhalt (Wert). Alle Schlüssel befinden sich im selben 160bit-Nummernraum wie die Kad-Ids der Teilnehmer.

Ein Teilnehmerknoten V speichert folgendermaßen ein Schlüssel-Wert-Paar am richtigen Platz im Kademlia-Netzwerk:

1. V lokalisiert die nächsten k-Knoten zum Schlüssel über die in Abschnitt 2.1.2 gegebene Knotensuche[8].
2. Mit einer sogenannten **Veröffentlichungs-Nachricht** von Teilnehmer V wird diesen k-Knoten das Schlüssel-Wert-Paar zum Speichern in ihrer sogenannten Hashtabelle übergeben. Somit ist das übergebende Paar mehrmals redundant im Netzwerk gespeichert.
3. Künftig kümmern sich die Knoten, die obiges Paar erhalten haben, automatisch um die weitere Verbreitung des Schlüssel-Wert-Paares. Wenn sich also ein neuer Teilnehmer dem Kademlia-Netzwerk anschließt und dieser noch näher am Schlüssel liegt, wird diesem Teilnehmer durch vorbezeichnete Verteilung das Schlüssel-Wert-Paar zur Speicherung mitgeteilt.

[7] Üblicherweise ist $\alpha = 3$
[8] Meistens wird k = 10 gesetzt

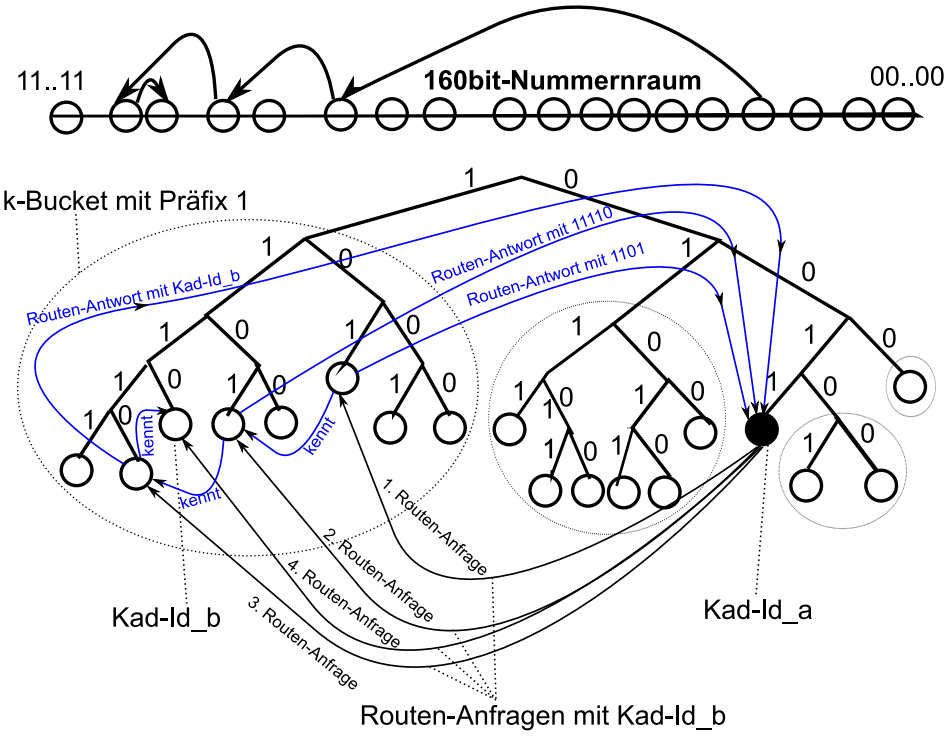

Abbildung 2: Routingmechanismus anhand einer Knotensuche. Lokation eines Knotens mit Präfix-Id 1110. Startknoten hat Präfix 0011. Durch sukzessives Abfragen und Erlernen von in Frage kommenden Knoten wird der Knoten mit Präfix 1110 gefunden.

Nach dem folgenden Schema kann ein Informationsinhalt über dessen Schlüssel abgerufen werden:

1. Im ersten Schritt wird wieder eine Knotensuche nach dem verlangten Schlüssel durchgeführt, um die nächsten k-Knoten zum Schlüssel zu erhalten.
2. Falls einer dieser Knoten den angeforderten Schlüssel in seiner Hashtabelle hat, wird die Suche abgebrochen und der dazugehörige Schlüsselwert dem anfragenden Teilnehmer zurückgeliefert.

Durch den Speicher- und Ladeprozess von Schlüssel-Wert-Paaren ergibt sich die Datenstruktur der sogenannten verteilten Hashtabelle (**DHT**[9]).

2.2 Die wichtigsten Funktionsweisen des Kademlia-Protokolls

Zur Aufrechterhaltung der Kademlia-Netz-Dienste (DHT, Routing, Netzwerkstruktur) bietet das Kademlia-Protokoll für Teilnehmerknoten vier Anfragetypen. In Tabelle 1 ist eine Übersicht dieser Anfragen und deren wichtigsten Funktionsweisen kurz erläutert.

Damit ein neuer Client aktiv im Kademlia-Netz teilnehmen kann, muss ein Kommunikationsvorgang mit bereits integrierten Peers hergestellt werden. Dazu muss der Client C einen **Bootstrap**-Vorgang wie folgt durchlaufen:

1. C generiert sich seine eigene Kad-Id über eine festgelegte Hashfunktion.
2. An die initial vorgegebenen Kontakte sendet der Client C eine sogenannte **Hello-Nachricht** mit seiner Kad-Id und IP-Adresse.

[9] distributed hash table (engl.)

Anfrage	**Funktionsbeschreibung**
Hello-Nachricht	ermittelt, ob ein Peer aktiv ist und informiert andere Peers über eigene Aktivität mit Tupel aus Kad-Id und IP-Adresse
Routen-Anfrage u. -Antwort	sucht Peers in der Nähe der übergebenen Kad-Id. Als Antwort wird eine Menge mit Tripeln der Form (IP, Port, Kad-Id) zurückgeliefert
Veröffentlichungs-Nachricht	speichert Schlüssel-Wert-Paar in DHT ab
Such-Anfrage u. -Antwort	erlaubt die Stichwortsuche nach Dateien über DHT. Als Antwort wird der Schlüsselwert aus DHT oder falls auf dem Peer kein solcher Schlüssel gespeichert ist, eine Menge von Tripeln mit (IP, Port, Kad-Id), die zur anfragenden Schlüssel-Id als nächstes liegen, zurückgeliefert

Tabelle 1: Übersicht der wichtigsten Funktionen innerhalb des Kademlia-Netzwerks

3. Als Antwort erhält C neue Kontakte, die innerhalb des Kademlia Nummernraums liegen. Diese Kontakte speichert er nach Algorithmus 2.1.2 in seinen k-Buckets.
4. Um nun seine nächsten k-Nachbarn kennenzulernen, führt Client C eine Knotensuche auf seine eigene Kad-Id aus. Dadurch registriert sich Client C als **Peer** im Netzwerk.

2.3 Anforderungen an File-Sharing Peer-to-Peer Systeme

Die im Vordergrund stehenden Anforderungen beim Design eines File-Sharing-Netzwerks sind die Datenkonsistenz und gute Performance innerhalb einer fehleranfälligen Umgebung [MaMa02]. Daraus sind folgende Probleme in Hinblick auf die Sicherheit solch eines Systems ersichtlich:

- Wie kann ein Benutzer eines Peer-to-Peer-Netzwerks sicher sein genau das Dokument zu erhalten, welches er angefordert hat?
- Wie verhält es sich mit der Verfügbarkeit des Dienstes bei der Durchführung eines D-DoS (Distributed Denial of Service) Angriffs?

Außerdem stellt sich die Frage, ob ein Benutzer in seiner Privatsphäre verletzt wird. Um die offenen Fragen zu behandeln, wird im nächsten Kapitel auf zwei wirkungsvolle Attacken näher eingegangen.

3 Angriffe auf Kademlia-Netzwerke

Die Hauptfunktionsweisen von Kademlia sind über Kapitel 2 bekannt und können jetzt durch verschiedene Möglichkeiten angegriffen werden.

3.1 Sybil-Angriff

Ziel des Sybil-Angriffs [Douc02] ist einen Abschnitt oder sogar das komplette Peer-to-Peer-Netzwerk durch einen Angreifer zu überwachen. Somit kann der Angreifer den Datenstrom von Netzteilnehmern im gewählten Netzabschnitt mitlesen. Der Datenstrom kann eine Such- oder Veröffentlichungs-Anfrage sein. Hierbei verwendet der Angreifer zur Durchführung dieses Angriffs einen einzigen Rechner mit modifizierten Peers und eine hohe Breitbandanbindung zum Internet. Die genannten modifizierten Peers unterscheiden sich in ihrer Häufigkeit an gesendeten Hello-Nachrichten und in ihren speziell gewählten Kad-Ids von den regulären Peers. Diese modifizierten Peers nennen sich hier Sybils.

Die folgende Übersicht gibt einen ersten Einblick in den Ablauf des Sybil-Angriffs:

> Zuerst werden die vorbezeichneten Sybils geschickt in das zu kompromittierende Netzwerk platziert. Wenn es dort den Sybils gelingt sich vollständig in das entsprechende k-Bucket der Routing-Tabellen normaler Peers einzutragen, dann müssen die Peers bei einer Anfrage in den kompromittierten Netzabschnitt mit den Sybils kommunizieren. Gerade diese Kommunikation wird von den Sybilknoten erwartet, weil es dann möglich wird in die Privatsphäre des normalen Peers einzubrechen oder darauf aufbauend, weitere Attacken zu starten.

Als Beispiel beobachtet ein Angreifer einen 8bit-Abschnitt des Kad-Netzwerks. Ein 8bit-Abschnitt entspricht die Beobachtung über alle Peers mit gleichem 8-stelligen Präfix. Es handelt sich also nach Kapitel 2.1.1, um einen Teilbaum des kompletten Kademlia-Präfixbaums. Der Angreifer wählt einen relativ kleinen Abschnitt, weil so die Netzwerklast und das Datenvolumen[10] begrenzt bleibt. Er geht in diesem Beispiel nach M. Steiner et-al. [StENB07] genau wie folgt vor:

> Ein Crawler[11] lernt, wie in Abbildung 3 dargestellt, auf dem Angreifer-Rechner die aktiven Peers im 8bit-Abschnitt kennen. Sein Rechner befindet sich bereits in diesem Abschnitt. Dazu führt sein Crawler eine Breitensuche im Präfix-Baum des 8bit-Abschnittes durch gezielte Route-Anfragen an Teilnehmer aus.

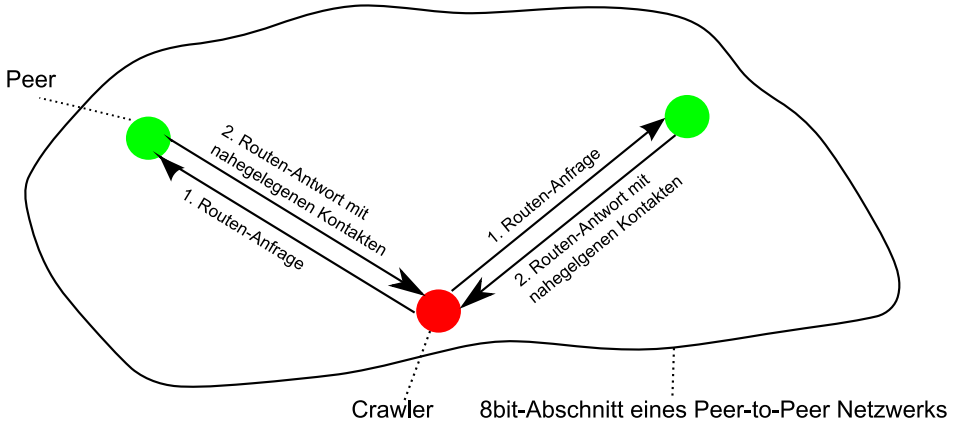

Abbildung 3: Crawler analysiert Netzwerk und erlernt neue Kontakte aus einem 8-bit-Netzabschnitts.

> Dann werden 2^{16} Sybil-Knoten mit gleichem 8-stelling Präfix, aber unterschiedlicher Kad-Id erstellt und ihre k-Buckets jeweils mit den gewonnen Peer-Kontakten vom Crawler gefüllt. Diese Sybil-Knoten senden, wie in Abbildung 4 gezeigt, Hello-Anfragen zu

[10] Alle Sybils verursachen einen einkommenden Datentransfer von ca. 400 KByte/s und einen ausgehenden Datentransfer von ca. 200 KByte/s. (vgl. [StENB07]).

[11] Laut M. Steiner et-al. [StENB07] kann ihr Crawler einen 8bit-Abschnitt innerhalb 2,5s und ein komplettes Kad-Netzwerk innerhalb etwa 8 Minuten ermitteln.

ihren bekannten Peers, um sich dort in die Routing-Tabelle einzutragen. Ein Peer wird nur dann Referenzen zu den vorbezeichneten Sybil-Knoten eintragen, wenn sein entsprechendes k-Bucket seiner Routingtabelle nicht belegt ist.

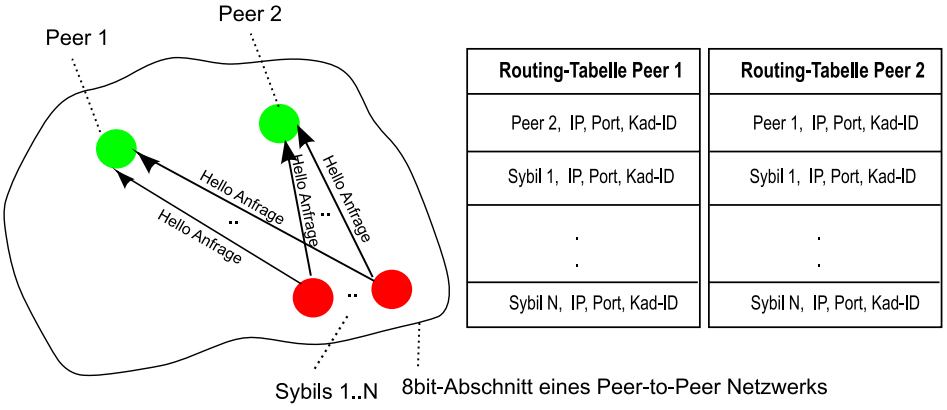

Abbildung 4: 2^{16} Sybilknoten modifizieren durch Hello-Anfragen die Routingtabelle der Peers innerhalb eines 8bit-Abschnittes.

Wird später ein normaler Peer, wie in Abbildung 5, eine Route-Anfrage mit beliebiger Kad-Id an Sybil-Knoten 1 stellen, antwortet dieser Sybil mit allen Kad-Ids aus dessen bekannter Sybil-Menge. Diese Sybils müssen im Sinne der XOR-Metrik näher zur angefragten Kad-Id sein. Andernfalls ignoriert der Sybil-Knoten 1 die Anfrage. Auf diese Art entsteht bei dem anfragenden Peer der Eindruck, das Ziel zu erreichen. Der Angreifer kann sich sicher sein, dass von diesem Peer eine Veröffentlichungs- oder Suchanfrage zu seinem Sybil-Knoten folgt.

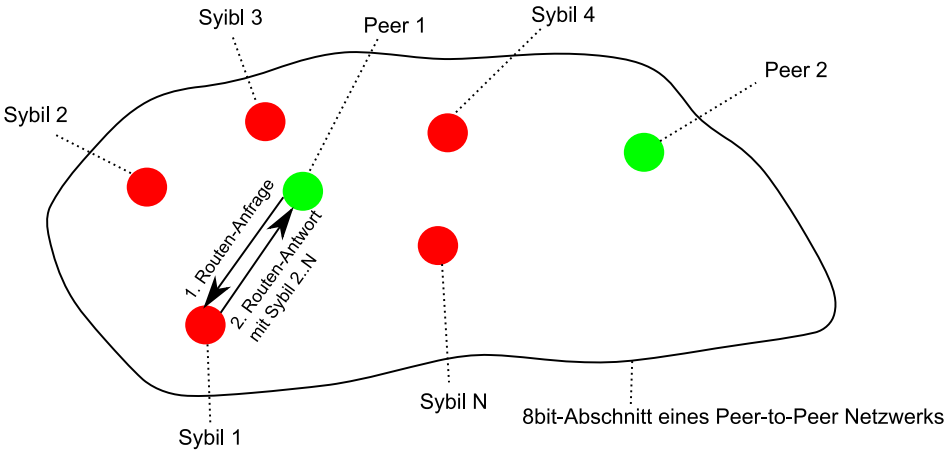

Abbildung 5: Sybil 1 beobachtet das Routing von Peer 1

Dadurch hat der Angreifer durch obiges Beispiel, die Möglichkeit nicht dem eigentlichen Kad-Protokoll zu folgen. Er kann eingehende Suchanfragen von anderen Teilnehmer abfangen. Die Sybils programmiert der Angreifer, so dass er jede Anfrage korrekt beantwortet, aber die einzelnen Nachrichten zu späteren Auswertung abspeichert.

Genauso verhält es sich mit veröffentlichten Datei- und Downloadanfragen eines Benutzers. Die Struktur der gewonnenen Information lässt sich in IP-Adresse, Port, Anfragetyp und Schlüsselinhalt gliedern. Somit können Rückschlüsse auf das private Verhalten des Nutzers in dem Kademlia-Netzwerk geschlossen werden.

Ausschließliches Sammeln von Benutzeraktivitäten ist nicht die einzigste Gefahr. Eine spezielle Form des obigen Sybil-Angriffs ermöglicht einem Angreifer im folgenden Beispiel einen spezifischen Stichwortschlüssel vom Kad-Netzwerk für andere Teilnehmer zu isolieren [StENB07]:

> Am Anfang platziert der Angreifer acht Sybil-Peers unmittelbar um den zu verschleiernden Stichwortschlüssel[12]. Dazu generiert er für jedes dieser Sybils, laut XOR-Metrik, eine entsprechend nahe Kad-Id zum Stichwortschlüssel.
>
> Jetzt senden die Sybils Hello-Anfragen zu den regulären Peers, um sich dort in die Routing-Tabelle einzutragen. Somit werden alle Routing-Anfragen - zum obigen Stichwortschlüssel - die Sybil-Peers des Angreifers erreichen und dort enden.

Analog stellte Peng Wang et-al. [WaTy08] auf eine andere Weise fest, Suchanfragen wie folgt ins Leere laufen zu lassen:

Basierend auf dem Kademlia-Protokoll zeigen die Autoren, dass der suchende Client bei einer maximalen Anzahl von 300 Suchantworten seine Suche einstellt[13]. Dadurch braucht ein Sybil genau 300 fälschliche Suchresultate zu liefern, um den Client den Zugriff auf eine Datei zu verwehren. Aber dieser Sybil muss schneller antworten, als ein regulärer Peer. Da dieser Angriff von der programmierten Client-Applikation abhängt, wurden in neueren Client-Applikationen (wie eMule, aMule) Sicherheitsmaßnahmen eingebaut, bei denen die zurückgelieferten Suchantworten zu dem Schlüssel der Suchanfrage passen müssen.

3.2 D-DoS

Der große Nachteil von Client-Server-Architekturen ist der „Single Point of Failure“. Das heißt, beim Angriff auf den Server werden seine verknüpften Dienstfunktionen lahmgelegt. Obwohl im Kademlia-Peer-to-Peer-Netz diese zentralen Funktionen über die Teilnehmer gemeinsam erbracht werden, hat ein Angreifer die Möglichkeit über eine sogenannte D-DoS[14]-Attacke das komplette System oder Abschnitte davon, außer Betrieb zu setzen. Üblicherweise ist ein Dienst außer Betrieb, wenn dieser keine Interaktion und angebotenen Leistungen mit dem Dienstnehmer erbringen kann. Technisch gesehen tritt der obige Fall ein, wenn entweder alle UDP-Verbindungen des Dienstes belegt sind oder die Bandbreite des Dienstsystems erschöpft ist.

Gegenstand im Folgenden werden sowohl die Überlastung des eigentlichen Peer-to-Peer-Netzwerks (Netzwerk ist Diensterbringer) als auch Ziele außerhalb des Systems, wie ein üblicher Webserver (Server ist Diensterbringer), sein. Gegliedert ist dieser Abschnitt in drei Ausprägungen von D-DoS-Angriffen:

1. Der klassische D-DoS-Angriff gegen Nutzer eines Peer-to-Peer-Netzwerks
2. Die Churn-Attacke [StRe06]
3. Der D-DoS-Angriff über Kademlia-Teilnehmer gegen externe Ziele

3.2.1 Klassischer D-DoS-Angriff

Der klassische Ablauf eines D-DoS-Angriffs gliedert sich in eine Vorbereitungsphase und eine tatsächliche Angriffsphase. Laut dem Bundesamt für Sicherheit (BSI) [BSI00] setzt ein Angreifer in der ersten Phase eine Client-Server-Architektur ein. Dazu installiert er eine Vielzahl von **D-DoS-Clienten** auf sehr vielen Netzwerkrechnern. Seine Clienten kontrolliert der

[12] Ein Experiment von M. Steiner et-al. [StENB07] zeigt, dass acht Sybil-Knoten ausreichend sind, alle Suchanfragen eines Schlüssels auf einen der Sybils zu terminieren

[13] Dies betrifft die Client-Applikationen aMule 2.1.3 und eMule 0.48a.

[14] distributed denial of service (engl.)

Angreifer über Befehle mit Hilfe eines zentralen Servers (ein sogenannter **D-DoS-Master**). Passiv warten diese Clienten auf die Aufforderung, ein Zielsystem in einem Kademlia-Netz anzugreifen. Der Angreifer erteilt die Befehle über seinen D-DoS-Master bei der Angriffsphase.

In den meisten Fällen versteckt der Angreifer seine D-DoS-Clienten auf verteilten PC-Systemen, die er zuvor über ein trojanisches Pferd infiziert hatte. Ihre Registrierung erfolgt völlig automatisiert bei dem zugehörigen Master-System.

Entschließt sich der Angreifer, wie in Abbildung 6 dargestellt, einen Teilnehmer im Kademlia-Netz anzugreifen, erteilt er seinem kontrollierenden D-DoS-Master das entsprechende Kommando zum Angriff. Daraufhin werden Informationen über das Ziel und weitere Befehlssequenzen zu allen D-DoS-Clienten versendet, die dann gemeinsam beginnen, DoS-Pakete über das Internet zu obigen Teilnehmer zu senden. Solche Pakete setzen sich aus assemblierten TCP-, UDP- und ICMP-Datenpaketen zusammen. Diese Daten verursachen beim genannten Teilnehmer eine Netzüberlastung. Damit erreicht der Angreifer, den Teilnehmer-Knoten vom Kademlia-Netz zu trennen. Somit sind dessen Dateien nicht mehr für andere Nutzer innerhalb Kademlia verfügbar. Allerdings bleiben die Metainformationen jener Dateien über die verteilte Hashtabelle noch für andere Teilnehmer im File-Sharing-Netzwerk abrufbar.

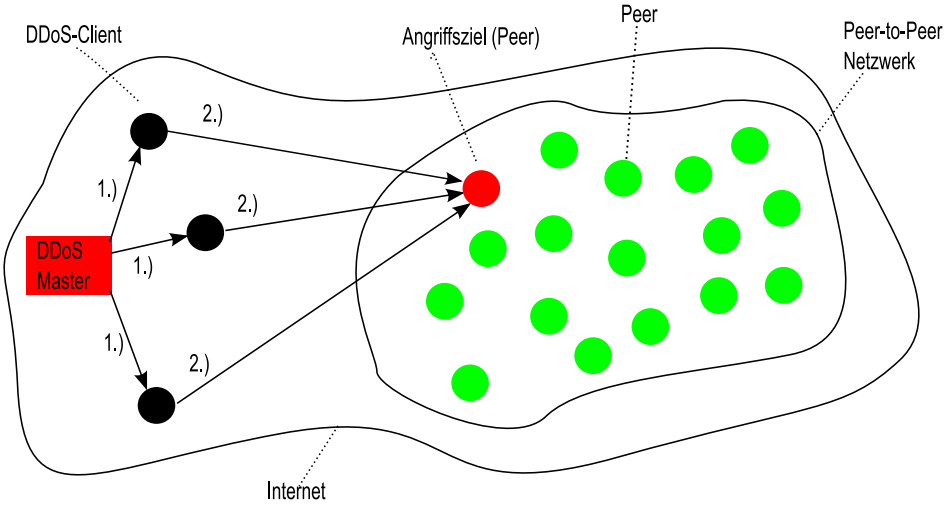

Abbildung 6: Beispiel eines klassischen D-DoS-Angriffs: 1.) Master erteilt Auftrag zum Angriff. 2.) Clienten greifen Ziel gemeinsam an.

Um eine gezielte Zerstörung der vorbezeichneten Metainformationen, über die Dauer der D-DoS Attacke zu erreichen, muss der Angreifer zusätzliche D-DoS-Ziele im Peer-to-Peer-Netzwerk bestimmen. Das heißt, er muss diejenigen Teilnehmer aus Kademlia kennen, welche jegliche Redundanzinformationen über die vorgenannten Dateien in ihrer DHT-Tabelle speichern. Der Angreifer kann ohne eine aktive Teilnahme im File-Sharing-Netzwerk diese nicht in Erfahrung bringen.

3.2.2 Churn-Angriff

Der Churn-Angriff [StRe06] ergibt sich gegen ein Peer-to-Peer-Netzwerk, wenn ein Angreifer sich wiederholt mit vielen Peers in das Netzwerk einklinkt und es unmittelbar darauf wieder verlässt.

Um eine Verbindung zum Netzwerk herzustellen, nutzen die Peers des Angreifers den Bootstrap-Vorgang (erläutert in Kapitel 2.2). Durch diesen Vorgang werden viele Kommunikationsnachrichten (Hello-Nachrichten, Routen-Anfragen) an teilnehmende Peers im Netzwerk

gestellt. Währenddessen speichern die vorbezeichneten Teilnehmer-Peers die Angreifer-Peers in ihrer Routing-Tabelle ab. Jedoch werden die Routingeinträge beim Verlassen der Angreifer-Peers ungültig. Wenn also ein Angreifer obigen Prozess schnell genug wiederholt, bewirkt er eine Belastung des Peer-to-Peer-Netzwerks und stört das Routing.

3.2.3 Der D-DoS-Angriff über Kademlia-Teilnehmer gegen externe Ziele

Die Grundidee dieses Angriffs ist mit Hilfe von vielen Teilnehmern aus dem Kademlia-Netz, ein externes System außerhalb Kademlia anzugreifen. Die Teilnehmer wissen nicht, dass sie zu einem D-DoS-Angriff benutzt werden.

Dafür muss ein Angreifer folgende drei Ziele erreichen:

1. Sybils ins Kademlia-Netz einschleusen
2. Routen-Antworten der Sybils mit IP-Adresse des D-DoS-Zielsystems modifizieren
3. Routen-Anfragen schneller als ein regulärer Teilnehmer beantworten

Der Angreifer platziert zuerst, wie im Sybil-Angriff (siehe Kapitel 3.1), seine Sybils im Kademlia-Netzwerk.

Danach warten - laut P. Wang et-al. [WaTy08] - die Sybils des Angreifers auf aktuelle Routen-Anfragen aus dem Peer-to-Peer-Netzwerk. Bei Eingang einer solchen Anfrage wird wie folgt beim entsprechenden Sybil-Knoten eine Routen-Antwort erzeugt:

> Die Antwort setzt sich aus 30 Kontakten zusammen. Bei jedem Kontakt wir das Tripel aus (IP-Adresse, UDP-Port, Kad-Id) durch den Sybil-Knoten modifiziert, indem das Tripel durch (IP-Adresse des D-DoS-Zielsystems, UDP-Port des Ziels, Routen-Anfrage-Schlüssel) ersetzt wird.

Wenn die regulären Peers die obigen modifizierten Antworten erhalten, versuchen sie unmittelbar mit den ersten α neuen Kontakten Verbindung aufzubauen[15]. Die vorbezeichneten regulären Peers verbinden sich zu diesen α-Kontakten, weil alle Kad-Ids aus den α-Kontakten identisch mit dem Routen-Anfrage-Schlüssel sind.

Allerdings befindet sich die gegebene IP-Adresse der Kontakte außerhalb des Peer-to-Peer-Netzwerks. Somit spricht das entsprechende Endsystem nicht das Kademlia-Protokoll und wird nicht antworten. Also wird die Time-out-Bedingung[16] des regulären Peers eintreten und in bestimmten Zeitabständen zu den nächsten α-Kontakten Verbindung aufbauen.

Der Angreifer überlastet das D-DoS-Zielsystem, wenn es ihm gelingt, ausreichend reguläre Peers mit obigen Verfahren zu gewinnen.

Dieser Angriff misslingt genau dann, wenn vor der Routen-Antwort der Sybil-Peers ein reguläre Peer korrekt antwortet.

Peng Wang et al. [WaTy08] demonstrierten sowohl theoretisch als auch durch Experimente, dass eine Internetanbindung mit einer Bandbreite von 100 Mbps ausreicht, diesen Angriff durchzuführen. Im Gegensatz dazu wird eine klassische verteilte D-DoS-Attacke zur selben Anzahl von Zielsystemen ungefähr 1 Tbps Bandbreite bei angenommener durchschnittlicher Downloadrate von 1 Mbps benötigen.

[15] In den meisten Kademlia-Implementierungen ist $\alpha = 3$.
[16] Bedingung für eine Wartezeitverletzung.

4 Fazit und Schlussbemerkung

Peer-to-Peer File-Sharing-Netzwerke lösen zwar Probleme der robusten Funktionsweise, effizientes Auffinden von verteilt gespeicherter Information und Skalierbarkeit bezüglich Anzahl der Benutzer. Jedoch haben die Angriffsszenarien gezeigt, dass ein reibungsloser Betrieb und die Privatsphäre eines Nutzers innerhalb des verteilten Systems gestört werden. Also können existierende File-Sharing Netzwerke, die das Kademlia-Protokoll einsetzen, nicht als sichere Netzwerke bezeichnet werden.

Obwohl der große Nachteil des „Single-Point of Failures“ (3.2) einer Client-Server-Architektur bei dezentralen File-Sharing Netzwerken nicht existiert, reicht eine Bandbreite von 100 Mbps zum Internet aus, um diese Schwäche größtenteils mit den gezeigten D-DoS-Methoden (aus Kapitel 3.2) herzustellen.

Die in Kapitel 3.1 vorgestellten Techniken können die Angreifer unter anderem zur Analyse der Privatsphäre eines Nutzers anwenden oder einem Netzteilnehmer falsche Ergebnisse einer Stichwort-Suchanfrage zurückliefern, sowie das Routing des Netzwerks beeinflussen.

Interessanterweise beziehen sich die dargestellten Angriffe nicht nur auf das Kademlia-Netzwerk, sondern zielen auch auf andere verteilte Systeme. Es muss also in Zukunft untersucht werden, welche dezentralen Netzwerke noch von solchen Angriffsmethoden betroffen sind.

Literatur

[BaLP05] S. Balfe, A. D. Lakhani und K. G. Paterson. Trusted computing: providing security for peer-to-peer networks. In *Proc. Fifth IEEE International Conference on Peer-to-Peer Computing P2P 2005*, 31 Aug.–2 Sept. 2005, S. 117–124.

[BaTe04] Jason E. Bailes und Gary F. Templeton. Managing P2P security. *Commun. ACM* 47(9), 2004, S. 95–98.

[Bidg03] Hossein Bidgoli. *Handbook of Information Security - Volume 1.* Wiley. 2003.

[BSI00] BSI. *Distributed Denial of Service (DDoS) Analyse der Angriffs-Tools*, 2000. `http://www.bsi.bund.de/fachthem/sinet/gefahr/toolsana.htm`.

[CavR05] Miguel Castro und Robbert van Renesse. *Peer-to-Peer Systems IV.* Springer. 2005.

[ChWC05] Nicolas Christin, Andreas S. Weigend und John Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, New York, NY, USA, 2005. ACM, S. 68–77.

[DeDa04] Prashant Dewan und Partha Dasgupta. Pride: peer-to-peer reputation infrastructure for decentralized environments. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, New York, NY, USA, 2004. ACM, S. 480–481.

[Douc02] John R. Douceur. The Sybil Attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London, UK, 2002. Springer-Verlag, S. 251–260.

[DrKR02] Peter Druschel, Frans Kaashoek und Antony Rowstron. *Peer-to-Peer Systems.* Springer. 2002.

[JoMW08] M. E. Johnson, D. McGuire und N. D. Willey. The Evolution of the Peer-to-Peer File Sharing Industry and the Security Risks for Users. In *Proc. 41st Annual Hawaii International Conference on System Sciences*, 7–10 Jan. 2008, S. 383–383.

[KaAG06] Andrew Kalafut, Abhinav Acharya und Minaxi Gupta. A study of malware in peer-to-peer networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2006. ACM, S. 327–332.

[MaMa02] Petar Maymounkov und David Mazieres. *Peer-to-Peer Systems*, Kapitel Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, S. 53–65. Springer. 2002.

[MaSc07] Peter Mahlmann und Christian Schindelhauer. *Peer-to-Peer Netzwerke.* Springer. 2007.

[Oram01] Andy Oram. *Peer to Peer: Harnessing the Power of Disruptive Technologies.* O'Reilly Media, Inc. 2001.

[ScMH08] J. Schafer, K. Malinka und P. Hanacek. Peer-to-Peer Networks Security. In *Proc. Third International Conference on Internet Monitoring and Protection ICIMP '08*, June 29 2008–July 5 2008, S. 74–79.

[ShSr06] Srinivas Shakkottai und R. Srikant. Peer to peer networks for defense against internet worms. In *Interperf '06: Proceedings from the 2006 workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems*, New York, NY, USA, 2006. ACM, S. 5.

[StENB07] Moritz Steiner, Taoufik En-Najjary und Ernst W. Biersack. Exploiting KAD: possible uses and misuses. *SIGCOMM Comput. Commun. Rev.* 37(5), 2007, S. 65–70.

[StKl05] Ralf Steinmetz und KlausWehrle. *Peer-to-Peer Systems and Applications.* Springer. 2005.

[StRe06] Daniel Stutzbach und Reza Rejaie. Understanding churn in peer-to-peer networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2006. ACM, S. 189–202.

[VoSh05] Geoffrey M. Voelker und Scott Shenker. *Peer-to-Peer Systems III.* Springer. 2005.

[WaTy08] Peng Wang und James Tyra. Attacking the Kad Network. In *Attacking the Kad Network*, 2008.

[ZhJK06] Bo Zhu, Sushil Jajodia und Mohan S. Kankanhalli. Building trust in peer-to-peer systems: a review. *Int. J. Secur. Netw.* 1(1/2), 2006, S. 103–112.

Abbildungsverzeichnis

Covert Channels in Onlinespielen

Benjamin Behringer

Covert Channels versuchen, einen unerkannten Datenaustausch zwischen zwei Punkten zu ermöglichen. Obwohl diese Idee nicht neu ist, bleibt sie als Anwendung der Steganographie doch immer dort relevant, wo über längere Zeit Kommunikation zwischen Bekannten und Unbekannten auftritt. Sie bindet sich dabei an das jeweils verwendete Protokoll, und es wird versucht, dieses so unauffällig wie möglich zu verändern, um Nachrichten zu übermitteln. Die Wege sind dabei vielfältig, und je mehr verschiedene Informationen über einen Kanal wandern, desto mehr Möglichkeiten gibt es, seine eigenen darin zu verstecken. Die Übertragungsprotokolle von Onlinespielen eignen sich unter anderem deshalb dafür, und ebenso, weil sie eine gewünschte und konstante Kommunikation darstellen. Mit dem großen Wachstum der Spielegemeinde im Internet ist das Problem des unerkannten Datenaustauschs über diese Plattformen heute sogar zum Problem von Regierungen geworden.

1 Motivation

Mit unterschiedlichen Methoden versucht man seit jeher gewisse Informationen vor anderen zu schützen. Man kann sie manchmal einfach wegschließen oder vernichten, meist jedoch sind diese Informationen nur wertvoll, wenn sie auch wenigen anderen Personen zugänglich gemacht werden. Dabei reicht oft auch simple Verschlüsselung, so gut sie auch gemacht sei, nicht mehr aus. Denn selbst die Information über die Kommunikation zweier Parteien kann eine wertvolle und kritische Information sein. Auf der Suche nach Möglichkeiten, Informationen auch unerkannt zu übertragen, existieren viele Wege. Augenzwinkern oder Mienenspiel als Beispiele alltäglicher versteckter Kommunikation, tote Briefkästen oder die Reihenfolge von Werbeblöcken im Fernsehen als weitere Möglichkeiten, die vielleicht ein bisschen die Vielfalt der Methoden widerspiegeln. In der heutigen Onlinegesellschaft kommt man aber bei der Suche nach dem Übertragungsmedium schnell auch auf das Internet. Leider ist die Vorstellung, alleine die Kommunikation über das Internet würde die Informationen vor fremdem Zugriff schützen, vollkommen falsch. Die Struktur des Internets, sogar sein Design, spricht genau für das Gegenteil. Also muss man hier, wie in der realen Welt, Wege finden, um seine Informationen sicher und eventuell versteckt zu übertragen. Auf der Suche nach einem "belebten" Platz im Internet, an dem man seine Informationen eventuell in der Anonymität der Masse weitergeben kann, trifft man schnell auf Onlinespiele. Deren Communities haben Millionen von Mitgliedern (World of Warcraft: über 11 Millionen [Odak08], Counterstrike: fast 3 Millionen [Verb06], etc...) und leben davon, dass so viele Spieler wie möglich gleichzeitig gemeinsam spielen können. Ihre Eignung und Nutzung als Medium zur versteckten Kommunikation soll im Folgenden betrachtet werden.

2 Parallelen zur Kryptografie und Grundlagen

2.1 Motivation

Wie bereits erwähnt, reicht es manchmal nicht aus, Informationen für Außenstehende unleserlich zu übertragen, sondern die Existenz der Kommunikation selbst soll verheimlicht werden. Hierfür ist das Internet auf den ersten Blick ein eher schlechtes Medium. Jedes vom Sender abgeschickte Paket kann auf jeder Station des Wegs gelesen, Sender und Empfänger zumindest für Teilstrecken des Wegs eindeutig bestimmt werden. Das Problem ist in der Kryptografie schon lange bekannt und wird sehr gut vom "Prisoner Problem" von G. J. Simmons [Simm83] beschrieben.

2.2 Das Prisoner Problem

Zwei Gefangene in einem Gefängnis, Alice und Bob, wollen einen gemeinsamen Fluchtplan erarbeiten. Es ist ihnen erlaubt, per Brief zu kommunizieren, jedoch werden alle Nachrichten von einem Aufseher, Wendy, kontrolliert. Offensichtliche Absprachen zur Flucht werden natürlich aussortiert, genauso wie verschlüsselte und anders verdächtige Briefe. Wird außerdem ein verdächtiger Brief gefunden, werden Alice und Bob in Einzelhaft gesteckt, was jegliche Flucht verhindert.

2.3 Folgerungen

Alice und Bob brauchen also eine Methode, ihre geheimen Nachrichten in den erlaubten Nachrichten zu versenden. Die erlaubte Nachricht mit der enthaltenen Geheimen darf dabei nicht von einer "normalen" Nachricht zu unterscheiden sein. Außer der Möglichkeit, Nachrichten auszutauschen, brauchen Alice und Bob dafür aber noch das gemeinsame Wissen über die hierfür verwendete Methode und den benutzten Schlüssel. Das Wissen hierüber kann nicht über eine normale Nachricht ausgetauscht werden, da es nicht versteckt werden kann. Es muss also vor dem Versenden der ersten geheimen Nachricht beiden bekannt sein.

2.4 Die Lösung

Die Lösung für dieses Problem liefert uns die Steganographie. Wir beschreiben damit eine verdeckte Kommunikation, derer sich nur Alice als Sender und Bob als Empfänger oder umgekehrt bewusst sind. Alle Anderen, im speziellen Wendy, die die Kommunikation überwacht, bemerken zwar die offentsichtliche Kommunikation, jedoch nicht die versteckte. Für Alice und Bob ist also entscheidend, wie gut die benutzte Methode der Steganographie ist, um nicht entdeckt zu werden.

Als Definition [Kalk07]:

> We define steganography as the practice of undetectably altering a Work to embed a message.

In dieser Definition wird auch klar, dass die Steganographie kein Verfahren ist, dessen Möglichkeiten beim Verstecken von Nachrichten in stetiger Kommunikation enden. Stattdessen kann, wie in der Definition beschrieben, in jeder beliebigen Arbeit eine versteckte Nachricht enthalten sein. Jedoch ist für die Betrachtung von versteckter Kommunikation über das Internet die wiederkehrende Kommunikation interessanter. Sollte man das Internet für versteckte Kommunikation nutzen, so werden meistens Geschwindigkeit und die Möglichkeit einer

schnellen Antwort eine Rolle spielen. Deswegen wäre eine Methode welche die gegenseitige Kommunikation ermöglicht vorzuziehen.

Hat man solch eine Methode gefunden, wendet man sie mit der geheimen Nachricht und dem vorher bekannten Schlüssel auf die unverfängliche Nachricht an. Man erhält dann, je nachdem wie gut die verwendete Methode ist, eine unverfängliche Nachricht, die die geheime Nachricht enthält.

Abbildung 1 zeigt dies, sei "M" die besprochene Methode.

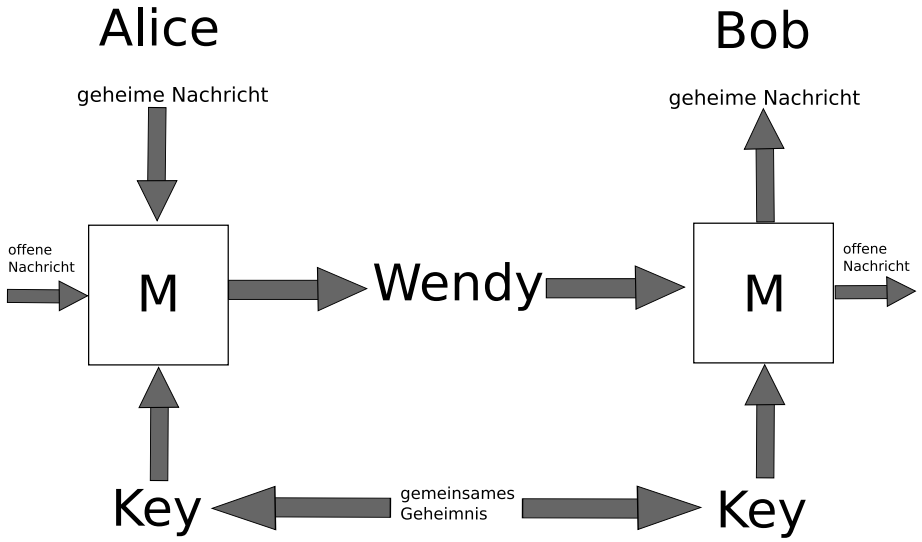

Abbildung 1: Das erweiterte Prisoner Problem

2.5 Ein Covert Channel

Als normalen Übertragungskanal (Channel) bezeichnet man im obigen Beispiel die offene Kommunikation zwischen Alice und Bob. Sie ist sowohl für diese beiden, als auch für Wendy sichtbar. Ein versteckter Übertragungskanal (Covert Channel) ist hier das Verschicken der durch die Steganographie-Methode geschützten geheimen Nachricht. Sie ist für Alice und Bob sichtbar, für Wendy jedoch transparent.

2.6 Störungen des Channels

Obwohl man den den Channel in der Theorie als störungsfrei betrachten kann, muss man bei der Anwendung damit rechnen, dass einzelne Nachrichten verändert oder gelöscht werden können. Dies kann man als Verhalten des Aufsehers Wendy abstrahieren. Wendy gilt als passiv, falls sie Nachrichten entweder nur durchlässt oder verwirft. Nach unseren Voraussetzungen verwirft Wendy eine Nachricht, sobald in dieser verbotene Kommunikation entdeckt wurde. Wendy gilt als aktiv, wenn sie die Nachricht verändert. Dies kann eine Vorsichtsmaßnahme zur Verhinderung von versteckter Kommunikation sein. Als bösartig gilt Wendy, wenn sie versucht, durch selbst eingebrachte Nachrichten Alice und Bob der geheimen Kommunikation zu überführen.

3 Bisherige Anwendungen

Schon lange vor dem Auftreten des Internets wurden mit Hilfe der Steganographie Covert Channels aufgebaut. Das "Prisoner Problem" ist nicht unbedingt aus der Luft gegriffen, ähnliche oder ganz anders geartete Anwendungen findet man in der Fachliteratur. Für uns von Interesse sind jedoch nur solche Covert Channels, die zum einen auf vorhandene Protokolle aufsetzen, und zum anderen einen im Internet vorhandenen Channel nutzen. Diese Covert Channels sind deshalb so interessant, weil sie einen ähnlichen Weg gehen wie Covert Channels in Onlinespielen. Diese nutzen auch einen vorhandenen Channel im Internet, nämlich die Verbindung der Spieler zum Server, sowie ein Protokoll, das vom System zum Austausch der Spieldaten verwendet wird.

Das Internet bietet eine Vielzahl von Protokollen, und alle entsprechen unseren Voraussetzungen. Hier sollen nur ein paar Beispiele betrachtet werden, um den Rahmen, in dem sich Covert Channels in Onlinespielen befinden, zu verdeutlichen.

3.1 Covert Channels in TCP/IP

Eins der am meisten benutzten Protokolle ist TCP/IP. Eine Arbeit von Murdoch und Lewis [MuLe05] befasst sich mit der Möglichkeit, in diesem Protokoll einen Covert Channel unterzubringen. Ihr Ansatz ist, die modifizierten Daten im Header des Protokolls unterzubringen. Dabei verwenden sie Felder, die meist mit zufälligen Daten initialisiert werden, oder bei denen zumindest für den Betrachter von außen (Wendy) nicht nachvollziehbar ist, wie die Werte zustande kamen. So ist es Wendy, selbst falls sie sich der Existenz des Covert Channels bewusst ist, nicht möglich, eine normale Nachricht von einer veränderten zu unterscheiden. Da die Mechanismen zum Erstellen der beiden Felder in vielen Betriebssystemen bekannt sind, entwickeln Murchoch und Lewis eine Abwandlung dieser Methoden am Beispiel von Linux und OpenBSD. Ihre Arbeit zeigt, dass allzu offensichtliche Veränderungen der Felder von Wendy erkannt werden können. Das Paper beschreibt, wie Covert Channels in TCP/IP Paketen eingebettet werden können und bietet Raum für weitere Entwicklungen. Es beschäftigt sich jedoch nur mit dem idealen Übertragungsfall, nämlich, dass von Alice versendete Pakete ohne Störungen Bob erreichen. Die in 2.6 beschriebenen Arten der Veränderung der gesendeten Nachrichten werden nicht diskutiert.

3.2 Covert Timing Channels

Anstatt die geheimen Botschaften in den offenen Nachrichten selbst zu verstecken, kann man natürlich auch der zeitlichen Abfolge der Nachricht eine eigene Bedeutung beimessen. Dieser Ansatz wird zum Beispiel im Paper von Berk, Giani und Cybenko untersucht [BGCH05]. Verschiedene Arten der Kommunikation über das Internet werden hier vorgestellt, und ihre Eignung für das zeitlich gestaffelte Senden erörtert. Im Vordergrund stehen dabei Informationsgehalt und Rauschfreiheit des Kanals. Die Autoren kommen zu dem Schluss dass ein möglichst kontinuierlicher Datenstrom, wie einen Multimedia-Stream, sich am Besten eignet, um durch die zeitliche Abfolge der Pakete eine Nachricht zu verschlüsseln. Insbesondere für die Datenrate des Covert Channels ist ein solcher Trägerstrom vorteilhaft, da ansonsten bei einem Trägerstrom mit niedriger Datenrate, wie beispielsweise einer SSH-Verbindung, nicht nur die Datenrate des Covert Channels leidet, sondern auch der Covert Channel leichter zu entdecken ist. Dieser Aspekt wird später noch einmal in 4 bei der Analyse von Onlinespielen als Träger für Covert Channels interessant. Auf die Erkennung von Covert Channels dieser Bauart wird ebenfalls eingegangen. Mithilfe der Statistik lassen sich Covert Channels, die auf Timing basieren, in manchen Fällen erkennen, auf jeden Fall ist die Erkennung aber ein

aufwendiger Prozess, der erst spät während der Kommunikation Aussagen über die Existenz eines Covert Channels machen kann, und dann auch nur mit einer gewissen Sicherheit, die von der Erkennungsmethode abhängt.

4 Eignung von Onlinespielen

Wenn man von Onlinespielen spricht, assoziiert man mit dem Begriff wohl am ehesten Spiele wie Counterstrike, World of Warcraft, Battlefield oder ähnliche. Aber auch Browser und textbasierte Onlinespiele fallen darunter. Fasst man den Begriff weiter, so gehören selbst Spiele wie Briefschach über E-Mail zur Kategorie "Onlinespiele". Solche Spiele sollen hier allerdings ausgeklammert werden, da erstens das eigentliche Spielgeschehen nicht an das Internet gekoppelt ist, man könnte zur Kommunikation hier auch wirkliche Briefe, Telefon, Rauchzeichen oder anderes verwenden. Zweitens ist der Kommunikation hier kein Rahmen gegeben, ein Brief der den nächsten Schachzug vorhersagt, kann, muss aber keine feste Form haben. Das Verstecken von Nachrichten in diese Art von Kommunikation gehört zwar auch zum Bereich der Steganographie, hat jedoch weniger mit dem Internet zu tun und ist deshalb hier uninteressant.

4.1 Das Protokoll

Für die restlichen Spiele gilt, dass die zeitliche Nähe der Aktionen der Spieler meistens viel größer ist. Dies setzt voraus, dass die Spieler, bzw. ihre Spielumgebung, regelmäßig mit dem neuesten Stand der Aktionen der Mitspieler versorgt werden. Damit man diese "Updates" sinnvoll verwenden kann, sollten sie einem im Spiel festgelegten Protokoll folgen, um vom Computer interpretiert werden zu können. Eine Interpretation der Updates durch den Spieler und die darauffolgende Eingabe in das Programm sind nicht nur fehlerträchtig, sondern machen die meisten Spiele auch aufgrund des Zeitaufwands unspielbar. Man kann also davon ausgehen, dass der Computer des Spielers in regelmäßigen Abständen versuchen wird, neue Spieldaten an die Mitspieler zu senden. Dies kann entweder in einem Peer-To-Peer System oder als Client-Server System geschehen. Viele heute angebotenen Onlinespielen verwenden dabei eine Client-Server Architektur.

Bei Onlinespielen, deren Spielablauf sehr schnell ist, profitiert der Spieler von einer niedrigen Antwortzeit (meist "niedriger Ping" genannt) des Servers. Da sich diese auch auf das Spiel auswirkt, und ein unregelmäßiger Ping im Spiel eventuell gut als ruckelnde Bewegung des Spielers sichtbar ist ("Lag"), sind Veränderungen im Timing der Kommunikation um einen Covert Channel aufzubauen sehr schwierig, wenn nicht sogar unmöglich.

4.2 Die Spieldaten

Die über das Protokoll verschickten Daten dienen dazu, die restlichen Spieler über Veränderungen in der Spielwelt zu informieren. Je komplexer diese Welt ist, und je mehr Interaktionsmöglichkeiten bestehen, desto größer wird die Datenmenge die für ein Update der Spielewelt übertragen werden muss. Man kann davon ausgehen, dass die Spieldaten nicht oder nur in einem gewissen Rahmen vorhersagbar sind. Die Spieler versuchen ja im Allgemeinen, im Spiel so gut wie möglich zu sein. Dabei entwickeln sie eigene Taktiken und Vorgehensweisen, die zu einem Vorteil führen sollen. Die übertragenen Spieldaten richten sich demnach nach der Vorgehensweise des Spielers und sind so individuell wie dieser selbst. Natürlich kann ein Spieler nicht die Regeln der Spielumgebung brechen, oder zumindest sollte dies dem Server oder

den anderen Clients auffallen. Dennoch bleiben die Spieldaten recht komplex und schwierig vorherzusagen. Außer den puren Spieldaten werden bei den meisten Spielen noch Texte verschickt, die im Spiel eine Chatkommunikation ermöglichen, oder andere Datenpakete zur Kommunikation, beispielsweise für Voice over IP. Da dieser Teil der Kommunikation jedoch nicht zum eigentlichen Spiel gehört, wird er hier ignoriert.

Da die Daten, wenn sich gerade kein Covert Channel in ihnen versteckt, im Prinzip nur reine Statusmeldungen sind, sind sie im Internet, außer für die Teilnehmer des Spiels, nur Daten von geringer Wichtigkeit. Sie unterliegen keiner offenen Filterung, wie beispielsweise E-Mails. Da die Kommunikation so schnell wie möglich gehalten werden soll, ist das Einfügen von Spam für alle beteiligten kontraproduktiv, da es die Datenlast unnötig erhöht. Das Einfügen von Daten in die Kommunikation, welche für das Spiel unnötig sind, scheidet also für das Etablieren eines Covert Channels von vorneherein aus, da es im normalen Verkehr stark auffallen würde.

4.3 Die Spieler

Die Teilnehmer eines Onlinespiels sind im Allgemeinen recht zufällig gewählt. Je mehr Personen ein Spiel im Internet spielen, desto geringer ist die Wahrscheinlichkeit, mit den gleichen Leuten ständig wieder zu spielen. Dies lässt sich natürlich durch die Wahl des einzelnen Spiels beeinflussen, und gilt nicht für private Spiele, die nur gewisse Teilnehmer zulassen. Wie eingangs erwähnt, ist die Zahl der Spieler von Spiel zu Spiel verschieden, geht aber eventuell bis in die Millionen.

Mitspieler lassen sich in drei Klassen aufteilen. Unbeteiligte, wohl die größte Gruppe, sind Spieler die nur zufällig am gleichen Spiel teilnehmen. Teilnehmer der geheimen Kommunikation bilden die zweite, kleinere Gruppe. Ihre Teilnahme am Spiel ist wahrscheinlich weniger zufällig, speziell bei vielen möglichen Spielen im Internet ist der versteckten Kommunikation zumindest eine Vorauswahl der Spiele, bzw. der Server, auf denen die Spiele laufen, sehr dienlich. Ebenfalls selten wird die dritte Spielergruppe auftreten. Diese nehmen, genau wie die Teilnehmer der versteckten Kommunikation, nur ihretwegen am Spiel teil, versuchen aber den Covert Channel aufzudecken oder abzuhören.

4.4 Hindernisse

Natürlich lässt sich die Kommunikation zwischen den Spielern nicht beliebig verändern. Neben den bereits erwähnten Schwierigkeiten mit der Veränderung des Timings und des Einfügens zusätzlicher Informationen bestehen weitere Probleme. Viele Onlinespiele schützen ihren Datenverkehr mit Verschlüsselung oder Prüfsummen. Manchmal werden auch externe Werkzeuge verwendet, um das Spieleprogramm auf Veränderungen hin zu untersuchen. Schafft man es, dies zu umgehen, kann man die Spieldaten eventuell nur minimal ändern, da die Änderungen ansonsten wieder auffallen würden. Ein weiteres Problem ist eher allgemeiner Natur und betrifft jede Art von Kommunikation. Das sogenannte “Rauschen” (das Verlorengehen von Daten, die Ankunft der Daten in falscher Reihenfolge oder das Auftreten zusätzlicher, ungewünschter Daten) beeinträchtigt sowohl die Kommunikation auf dem offenen, als auch die im Covert Channel. Ein Mechanismus zur Einbettung eines Covert Channels muss das Rauschen auf dem Channel ausgleichen können, andernfalls ist die Kommunikation eventuell nicht möglich.

4.5 Zusammenfassung

Onlinespiele scheinen ein interessantes Medium für Covert Channels zu sein. Neben der stetigen Kommunikation, der Zufälligkeit und Vielfalt der übertragenen Daten, sowie der relativen Unwichtigkeit der Übertragungen ist vor allem die große Zahl an Spielern interessant. Man hofft, dass die geheime Kommunikation in den großen Mengen irrelevanter Daten keine Spuren hinterlässt. Diese Faktoren sollten es ermöglichen, in Onlinespielen Covert Channels unterzubringen, trotz der zu überwindenden Hindernisse.

5 Arten von Covert Channels

Es gibt zwei bekannte Proof of Concept Implementierungen für Covert Channels in Onlinespielen. Die Arbeit von Zander, Armitage und Branch beschäftigt sich mit Covert Channels in First Person Shootern [ZaAB08], wohingegen die ältere Arbeit von Murdoch und Zieliński sich mit Covert Channels in einem 4 Gewinnt Spiel [MuZi04] befasst.

5.1 Covert Channels in Quake 3 Arena

Quake 3 Arena kommt für viele dem intuitiven Verständnis eines Onlinespiels recht nahe. Es ist sehr bekannt und eignet sich zum Entwickeln einer Proof of Concept Implementierung recht gut, da der Sourcecode des Spiels frei verfügbar ist. Der Ansatz von [ZaAB08] soll hier kurz beschrieben werden.

5.1.1 Spiel und Technik

Quake 3 Arena (Q3) ist ein Egoshooter, man steuert die Spielfigur in einer 3D-Welt aus der Egoperspektive. Um zusammen mit anderen Spielern zu spielen, tritt man einem Spiel bei, das auf einem Server irgendwo im Inter- oder Intranet läuft. Der Server und die Clients kommunizieren über UDP/IP. Falls der Administrator des Servers keine Ausnahmen definiert hat, kann jeder, der sich mit dem Server verbinden kann, am Spiel teilnehmen. Q3 ist ein beliebtes Spiel unter anderem auf LAN-Parties und hat eine große Fangemeinde.

Der Quellcode des Kommunikationsprotokolls von Quake 3 ist wie der des Spiels öffentlich verfügbar [Char05]. Er wurde über die Jahre analysiert und das Protokoll von verschiedenen Personen verständlich aufbereitet [Auri] [Darr]. In [Auri] wird auch erwähnt, dass das Protokoll große Ähnlichkeiten zu denen von Quake 2 und Half-Life aufweist. Jeder Client schickt in kurzen Zeitabständen (maximal einmal in 10ms) ein Update seiner aktuellen Spieldaten an den Server. Dieser schickt alle 50ms einen Snapshot aller ihm zur Verfügung stehenden, und für den jeweiligen Client relevanten, Spieldaten an die Clients, die dann mit diesen Daten ihre Spielwelt aktualisieren. Die von den Clients vollzogenen feineren Zwischenschritte werden verworfen. Die Zeit zwischen den Snapshots wird als Round Trip Time (RTT) bezeichnet. Der Server versendet die jeweils letzten vom Client bekannten Daten. Es ist klar, dass keiner der Clients ständig alle verfügbaren Daten erneut benötigt. Stattdessen werden nur die wirklich notwendigen Daten übertragen, also die, die sich auf den Spieler auswirken und seit dem letzten Snapshot geändert wurden.

5.1.2 Die Idee

Wie in 4.1 und 4.2 diskutiert, scheiden größere Veränderungen des Timings in der Kommunikation mit dem Server und den Nachrichten selbst aus. Man kann jedoch eine leichte

Veränderung der Spieldaten herbeiführen, die ebenfalls Information trägt. Diese Spieldaten dürfen natürlich vom Server nicht auf Veränderung überprüfbar sein. Es bietet sich hierfür zum Beispiel der Blickwinkel des Spielers an. Dieser setzt sich aus der Blickrichtung des Spielers relativ zu seiner Hochachse und seiner Querachse zusammen. Da diese Werte allein vom Spieler bestimmt werden, kann die Manipulation im einzelnen Snapshot nicht entdeckt werden. Alice schickt konstant die manipulierten Werte an Bob, der jedoch bekommt nur die Daten des jeweiligen letzten Snapshots vom Server. Da Alice häufiger an den Server sendet als dieser zurück, muss sie aufpassen, dass alle gesendeten Pakete die Informationen enthalten, die Bob benötigt. Sie wechselt die Information, sobald sie durch den Snapshot des Servers angezeigt bekommt, dass Bob die vorherige Information erhalten hat.

Abbildung 2 zeigt die relevanten Werte.

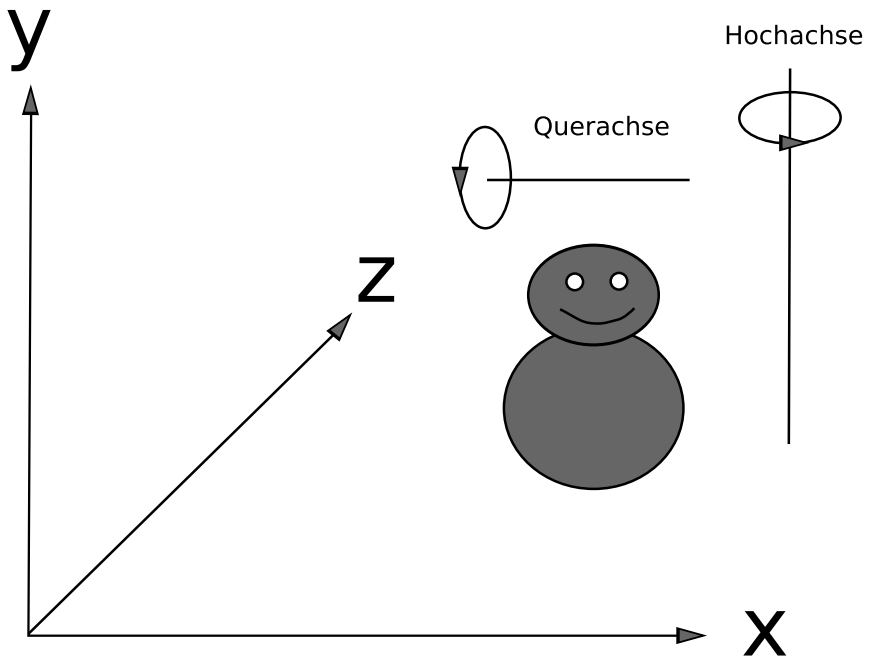

Abbildung 2: Die Blickachsen

Sei nun der Einfachheit halber die von Alice an den Server geschickte Position ein Skalar x_i, und die Position aus dem vom Server geschickten Snapshot y_i. Um eine Information zu übertragen, benutzt man den Unterschied $\Delta y = y_i - y_{i-1}$. Die Information b ergibt sich dann aus der Formel

$$b_i = (y_i - y_{i-1}) \mod 2^N \tag{1}$$

Da Alice nicht wissen kann, welche ihrer gesendeten Spieldaten vom Server als Snapshot benutzt werden, muss sie bis zum Erhalt des nächsten Snapshots alle Daten so verändern, dass sie die Information b_i tragen. Erhält Alice einen Snapshot, der ihr b_i enthält, so erhöht sie i um eins und beginnt b_{i+1} zu senden. Bis dahin wählt sie ihr

$$\bar{x}_i = x_i + \delta_i \tag{2}$$

Aus (1) und (2) folgt:

$$\delta_i = \begin{cases} b - (x_i - y_{i-1}) \mod 2^N & : \; x_i - y_{i-1} \geq 0 \\ -b - (x_i - y_{i-1}) \mod 2^N & : \; x_i - y_{i-1} < 0 \end{cases} \tag{3}$$

Lässt man das Rauschen des Channels außer Acht, kann Bob mit Gleichung (1) dann b wieder zusammensetzen. Man kann gut erkennen, wie der Channel arbeitet, wenn man ein paar Schritte durchführt, wie in Abbildung 3 nach [ZaAB08] zu sehen. Sei hierbei $N = 1$

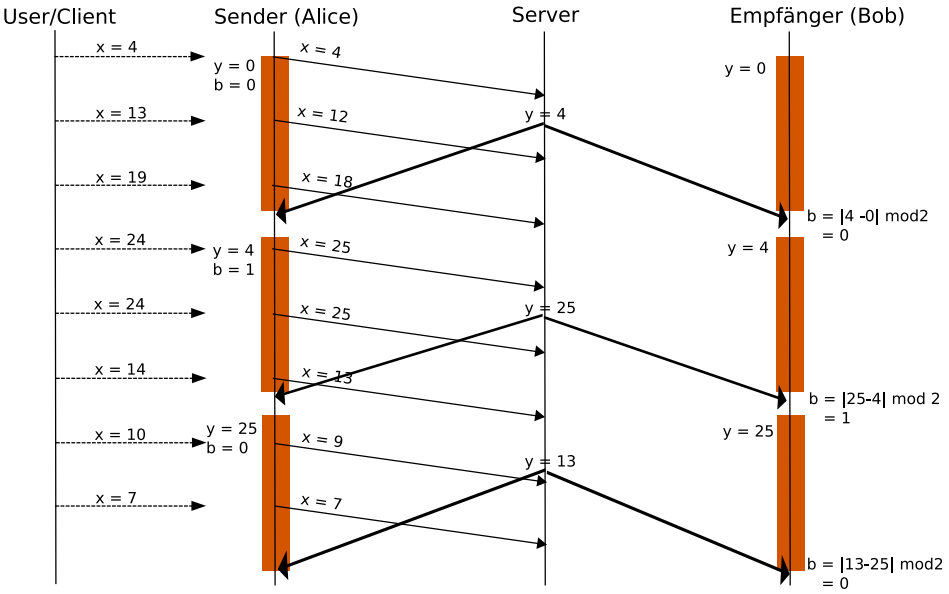

Abbildung 3: Eine vereinfachte Beispielkommunikation

Man sieht, dass Alice nicht ständig das gleiche x_i senden muss. Die von Alice gesendeten x_i müssen sich nur in der gleichen Äquivalenzklasse befinden, die vom Parameter N gegeben werden. Für $N = 1$ sendet Alice theoretisch bei jedem Snapshot ein Bit an geheimen Informationen an Bob.

Dies funktioniert, solange die Zeit zwischen dem Erhalt eines Snapshots und dem Absenden des nächsten beim Server groß genug ist, sodass Alice eine neue modifizierte Nachricht mit einer anderen Information verschicken kann. Ist dies nicht der Fall, so muss die Zeit für den Informationsfluss erhöht werden. Alice sendet dann zum Beispiel nur noch alle zwei Snapshots ein Informationspaket.

Dies war natürlich nur eine Abstraktion, man kann selbstverständlich mit beiden Blickwinkeln gleichzeitig arbeiten, und dabei die doppelte Menge an Informationen pro Snapshot verschicken. Natürlich müssen vorher Alice und Bob sowohl N, als auch die zu nutzenden Achsen und die Reihenfolge in der die Informationen aneinandergehängt werden bekannt sein.

5.1.3 Zuverlässigkeit

Um eine von Alice so gesendete Information empfangen zu können, muss Alice im Spiel für Bob relevant sein, ansonsten erhält Bob im Snapshot keine Informationen über Alice. Dies bedeutet meistens, dass Alice im Sichtbereich von Bob sein muss. Weiß Alice, welcher Spieler Bob ist, so kann sie gezielt nur dann Informationen verschicken, wenn sie weiß, dass sie im Sichtbereich von Bob ist (Unicast nach der Terminologie von [ZaAB08]). Ist sie sich der Identität von Bob nicht bewusst, muss sie die Informationen an jeden Spieler schicken, der sie in seinem Sichtbereich hat, in der Hoffnung dass Bob die Informationen erhält (Multicast, ebenfalls nach [ZaAB08]). Grundsätzlich ist der Unicast dem Multicast vorzuziehen, da die Übertragung auch stoppen kann, sobald Bob Alice nicht mehr sieht. Sendet Alice im Multicast an alle Spieler, so kann dies in einer großen Anzahl verlorener Informationen resultieren. Eine gestörte Verbindung zum Server bei einem der beiden kann ebenfalls zu verlorenen

Informationen führen, da eventuell Snapshots nicht interpretiert werden können. Um ohne Kenntnis der Identität von Bob trotzdem den Unicast verwenden zu können, wäre es möglich, so lange eine Sequenz bestimmter Länge im Multicast zu senden, bis Bob diese erhalten hat und mit einer weiteren bekannten Sequenz antwortet. Von da an wüsste Alice, wer Bob ist.

5.1.4 Gegenmaßnahmen

Man kann auf verschiedene Arten versuchen, den Covert Channel zu entdecken. Der einfachste Ansatz, das Teilnehmen am Spiel und die Suche nach verdächtigen Spielerbewegungen, wird jedoch nicht funktionieren, da die Änderungen der Spielerbewegung zu klein sind, um mit bloßem Auge erkannt zu werden. Erhöht man jedoch die Granularität der Spielerbewegung, sodass eine Veränderung sichtbar wird, ist dies ein grober Eingriff in das Gameplay. Eventuell wird das Zielen dadurch so eingeschränkt, dass das Spiel unspielbar wird.

Eine andere Variante ist, als Teilnehmer am Spiel zu versuchen Kommunikation auf dem Covert Channel selbst aufzufangen. Hierzu benötigt man jedoch Informationen über die Implementierung des Channels, ohne die eine Kommunikation nicht offensichtlich ist. Beispielsweise kann man im Vorhinein als Außenstehender nicht wissen, welche Achsen des Spielers verändert werden. So gehen einem eventuell beim Versuch die Kommunikation mitzuverfolgen, Informationen verloren oder man hat viele Störsignale.

Als Variante dieses Ansatzes bietet es sich an, das Spiel aufzuzeichnen und später mithilfe statistischer Methoden auf die Existenz eines Covert Channels zu überprüfen. Hat man die Möglichkeit die Kommunikation zwischen Server und Clients direkt am Server abzugreifen, stehen einem alle Spielerbewegungen zur Analyse zur Verfügung. Ohne Vorwissen über den Covert Channel ist dies wohl der erfolgversprechendste, aber auch der aufwändigste Ansatz.

Kann man den Covert Channel nicht entdecken, weiß aber um seine Existenz, kann man versuchen den Channel zu stören. Durch wiederum das Verändern der Spielerbewegungen in minimalen Umfang, aber um zufällige Werte, kann ein Rauschen in den Channel eingebracht werden, das die Kommunikation behindert. Jedoch können Alice und Bob mit dem Erhöhen der RTT in gewissem Maße kontern. Damit werden die b_i über mehrere Snapshots gesendet, um Störungen zu minimieren.

Es wird klar, dass der Covert Channel nur schlecht verhindert oder gestört werden kann, da die über das Protokoll stattfindende Kommunikation in ihrer Existenz und Präzision der Beschreibung der Spielsituation gewünscht ist und jeder Eingriff eine potentielle Störung des Spiels darstellt.

5.1.5 Implementierung

Die Referenzimplementierung benutzt das Covert Channels Evaluation Framework [Zand], ein Entwicklungswerkzeug zum Erstellen und Testen von Covert Channels. Um externe Tools, die die Integrität des Spiels überprüfen zu umgehen, setzt sich dieses Tool als Proxy zwischen Alice, Bob und jeweiligen Server. Diese Art der Modifikation kann nicht aufgespürt werden und kommt ohne Änderung am Programmcode des Spiels aus. Es würde Alice und Bob sogar erlauben, Nachrichten über den Kanal auszutauschen, ohne selbst am Spiel teilzunehmen. Jemand anders könnte für sie spielen. Dieser Ansatz könnte aber zu Problemen führen, sobald man eine verschlüsselte Kommunikation zwischen Client und Server hat und der Programmcode des Spiels nicht einsehbar ist.

5.1.6 Ergebnisse

Die Versuche des Teams von [ZaAB08] zeigen, dass bei einem Q3 Spiel eine versteckte Datenrate allein der gesendeten Daten von 14-18bit/s im Broadcast, bei einem Paketverlust von 43-55%, und eine Datenrate von 8-10bit/s bei einem Verlust von 3-5,5% im Unicast möglich sind. Damit liegen die Netto-Datenraten derer Informationen die tatsächlich bei Bob ankommen bei 7-9 Bit sowohl im Unicast als auch im Broadcast. Dies reicht für kleinere Textnachrichten oder Ähnliches. Dabei war der Covert Channel nicht für die Spieler mit bloßem Auge sichtbar, die Veränderungen der Positionsdaten waren so gering, dass man sie nicht wahrnehmen konnte.

6 Schlussfolgerung und Ausblick

Covert Channels in Onlinespielen sind eine interessante Anwendung der Steganographie und zeigen einmal mehr deren Mächtigkeit. Sie bieten die Möglichkeit einer versteckten Kommunikation über ein bekanntes und beliebtes Medium. Die große Spielerzahl und die Vielzahl der Spiele erlauben es, währenddessen unerkannt zu bleiben, quasi in der Masse unterzugehen. Obwohl diese Problematik mittlerweile auch von Regierungen erkannt wurde, und davor gewarnt wird [Wied08], stellen diese Aktivitäten jedoch nur, wie bereits erwähnt, eine Anwendung der Steganographie dar. Sie hat damit keinen höheren Stellenwert als sonstige Anwendungen dieser Wissenschaft. Jedoch wurden Covert Channels in Onlinespielen bisher mit wenig Aufmerksamkeit bedacht, und nicht annähernd so gut erforscht wie Covert Channels über andere Medien und Protokolle. Dies wird in Zukunft interessante Ansätze für die Forschung bieten.

Literatur

[Auri] Luigi Auriemma. Quake III 1.32 net information protocol 0.1.1. http://aluigi.altervista.org/papers/q3info.txt.

[BGCH05] V. Berk, A. Giani, G. Cybenko und NH Hanover. Detection of Covert Channel Encoding in Network Packet Delays. *Department of Computer Science, Dartmouth College, Technical Report TR2005536*, 2005.

[Char05] CharMe. Quake 3 Quellcode veröffentlicht. *http://www.spieleflut.de/news/15767/*, 2005.

[Darr] Darren. Quake 3 Network Protocol. http://www.tilion.org.uk/Games/Quake_3/Network_Protocol.

[Kalk07] Ingemar J. Cox Matthew L. Miller Jeffrey A. Bloom Jessica Fridrich Ton Kalker. *Digital Watermarking and Steganography Second Edition*, Kapitel 12, S. 425. Morgan Kaufmann Publish. 2007.

[MuLe05] S.J. Murdoch und S. Lewis. Embedding Covert Channels into TCP/IP. *LECTURE NOTES IN COMPUTER SCIENCE* Band 3727, 2005, S. 247.

[MuZi04] S.J. Murdoch und P. Zielinski. Covert Channels for Collusion in Online Computer Games. In *Proceedings of the Sixth Information Hiding Workshop, Canada*. Springer, 2004.

[Odak08] Sandro Odak. World of Warcraft Über 11 Mio. Spieler. *gamezone.de*, 2008.

[Simm83] Gustavus J. Simmons. The Prisoners' Problem and the Subliminal Channel. In *CRYPTO*, 1983, S. 51–67.

[Verb06] Markus Verbeet. Chucky muss trainieren. *www.spiegel.de*, 2006.

[Wied08] Annette Wieden. Terror-Aktivitäten in WoW? *http://www.buffed.de*, 2008.

[ZaAB08] S. Zander, G. Armitage und P. Branch. Covert channels in multiplayer first person shooter online games. In *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, 2008, S. 215–222.

[Zand] S. Zander. CCHEF - Covert Channels Evaluation Framework. Technischer Bericht 080530A, Centre for Advanced Internet Architectures, Swinburne University of Technology Melbourne, Australia.

Abbildungsverzeichnis

Zeitbasierte Cheats in Onlinespielen

Dominik Ristau

In kommerziellen Onlinespielen ist Cheating ein ernstzunehmendes Problem, für das hier einige Möglichkeiten zur Lösung vorgestellt werden. Die vorgestellten Protokolle versuchen zeitbasierte Cheats zu verhindern. Dazu wird zunächst ein stop-and-wait Protokoll eingeführt, anschließend wird dieses durch Modifikationen effizienter gemacht. Dann wird ein Peer-to-Peer Protokoll dargestellt, das mit Hilfe einer Art Abstimmung den Status der Mehrheit der Spieler miteinbezieht. Im vorletzten Abschnitt wird noch ein Ansatz gezeigt der nicht versucht die Cheats zu verhindern, sondern Cheater zu erkennen. Abschließend wird noch auf eine sich an die Situation anpassende Architektur eingegangen.

1 Einleitung

In Onlinespielen treffen sich eine große Zahl von Spielern, um an virtuellen Wettkämpfen teilzunehmen. Genauso wie es eine Vielzahl verschiedener Spiele gibt, findet man auch die unterschiedlichsten Arten von Cheats, die in Onlinespielen verwendet werden. Der Cheater ist damit meist so stark im Vorteil, dass der Spaß der anderen Spieler beeinträchtigt wird und diese nicht weiterspielen. Die vielen Varianten in denen Cheats auftreten können, macht es schwierig verlässliche Lösungen zu finden. Das Cheaten selbst hingegen gestaltet sich oft wesentlich einfacher, da diese in Communities verteilt werden oder sogar ganze Frameworks entstehen. Hinzu kommt, dass die steigende Anzahl von Spielern in großen virtuellen Welten das klassische Client/Server-Prinzip an wirtschaftliche Grenzen bringt. Deshalb scheint der Umstieg auf Peer-to-Peer-Architekturen hier sinnvoll. Dadurch erkauft man sich allerdings auch den Nachteil der dezentralen Informationsverwaltung, die das Spiel anfälliger fürs Cheaten macht. Die Skalierung ist bei schnelleren Spielen wegen der geringeren Spielerzahl weniger problematisch, allerdings sorgen hier schon kleinere Verzögerungen im Spielverlauf für einen gestörten Spielfluß und mindern damit den Spielspaß. Es muss also ein Kompromiss zwischen Sicherheit und Spielbarkeit gefunden werden.

Die folgenden Betrachtungen konzentrieren sich auf zeitbasierte Cheats. Diese kommen dadurch zustande, dass der Zeitpunkt des Absendens einer Nachricht von einem Client selbst bestimmt werden kann. Ein Cheater kann Veränderungen am Ablauf auf Protokollebene vornehmen und sich einen Vorteil erzwingen, wenn die Reihenfolge von Ereignissen wichtig ist. Er hat dann beispielsweise mehr Zeit zu reagieren oder kann die Bewegungen anderer Spieler früher als gedacht verwenden. Um dies zu verhindern werden Gegenmaßnahmen vorgestellt, die sich in ihrer Herangehensweise und Performanz unterscheiden.

2 Grundlagen

2.1 Begriffe

In diesem Abschnitt werden einige Begriffe und abkürzende Schreibweisen vorgestellt.

Die Ereignisse in einem Onlinespiel finden zu bestimmten Zeitschritten auf einem Client statt. Ein einzelner Zeitschritt wird *Frame* genannt. Ferner sei f_i^k der i-te Frame des k-ten Spielers. In vielen Echtzeitspielen ist ein flüssiges Erneuern der Frames eine Voraussetzung für die Spielbarkeit. Die Anzahl der Frames pro Sekunde wird Framerate, kurz: r genannt. Die Dauer eine Frames ist ein Maß für die Geschwindigkeit in der das Spiel voranschreitet und ist mit $1/r^k$ abhängig von der Situation in der sich das Protokoll gerade für den Spieler k befindet. Mit r_{max} ist die in Spielen meist festgelegte maximale Framerate gemeint, typischerweise ist sie bei 50 fps (frames per second).

Von Bedeutung in Onlinespielen ist auch die Verzögerung von einem Spieler p^i zu einem Anderen p^k, kurz: d_k^i. Analog bedeutet $d_{max} = \{max(d_i^j), (p^i, p^j) \in \mathcal{P}^2\}$, wobei $\mathcal{P}$ die Menge aller Spieler sein soll. Dabei nehmen wir an das $d_k^i = d_i^k$ für alle $p^i, p^k \in \mathcal{P}$.

Die beim Start eines Spiels beginnende zentrale Zeit, die unabhängig von der im Spiel aus der Sicht der Spieler vergangenen Zeit bzw. Anzahl an Frames ist, wird *Wallclock Time* genannt.

Ein Spieler p^k führt zu einem Zeitpunkt f_i^k eine Aktion aus und hat dann einen neuen Status (auch Update) s_i^k. Abkürzend wird im Folgenden auch oft $s_i^{j \neq k}$ geschrieben, womit die Status der anderen Spieler $p^j \in \mathcal{P} \backslash \{p^k\}$ gemeint sind. Eine Aktion kann beispielsweise bedeuten, dass er seine Geschwindigkeit drosselt, generell seine Position ändert oder Gegenstände ablegt.

2.2 Dead-Reckoning

Dead-Reckoning (zu dt.: für verloren halten) ist eine Technik die in Onlinespielen weite Verbreitung findet, vorallem in First Person Shootern (*FPS*). Wann immer ein Spieler während des Spielverlaufs keine Updates seines Spielstatus mehr sendet, wird aus dem vorherigen Status (z.B.: Position und Geschwindigkeit) unter einigen Annahmen (z.B.: Geschwindigkeit ist konstant) sein neuer Spielstatus angenähert. Dieser wird dann verwendet, solange bis die Updates des Spielers wieder ankommen. Dadurch versucht man den Einfluss einiger Spieler mit schlechter Verbindung auf den Spielfluß aus der Sicht der anderen Spieler zu minimieren.

In sehr schnellen Spielen wie FPS wird Dead-Reckoning oft als unverzichtbar dargestellt. Da es, auf Grund der hohen Frequenz in denen Nachrichten versendet werden, sofort ins Gewicht fällt, wenn einige davon verloren gehen.

Zwar verbessert Dead-Reckoning den Spielfluss, es bringt aber auch einige Schwierigkeiten mit sich, die oft als unfair empfunden werden. So kann ein Spieler, wenn für ihn gerade Dead-Reckoning angewendet wurde je nach Protokoll auf einmal an einem anderen Ort stehen. In Peer-to-Peer Architekturen kann es zu kritischen Situationen kommen, die schwer aufzulösen sind. Zum Beispiel wenn ein dead-reckoned Spieler einen anderen Spieler zerstört, dieser wiederum eine dritten, und dann die Updates vom dead-reckoned Spieler wieder ankommen.

2.3 Jitter

Mit Jitter bezeichnet man den Unterschied in den Verzögerungen in denen ein Paket im Verhältnis zu den Anderen ankommt. Ist der Jitter zu stark schwankend, dann kann es im Spiel zu stockendem Spielfluß kommen. Hierbei bleibt es dem Protokoll überlassen dies auszugleichen.

2.4 Cheat

Unter einem *Cheat* (zu dt.: *Schwindel, Betrug*) versteht man jedwedes Handeln eines oder mehrerer Teilnehmer des Spiels durch die sie sich Vorteile gegenüber anderen Spielern verschaffen, indem sie nicht nach den gegebenen Spielregeln oder nicht im Sinne des Betreibers

handeln. Ein Spieler könnte sich schneller bewegen, einen größeren Bereich einsehen oder bessere Gegenstände bekommen als Andere. Es gibt verschiedene Möglichkeiten wie ein Spieler cheaten kann, zum Beispiel:

- durch Veränderung der Spielesoftware (Client)
- durch das Abfangen, Ändern, Blockieren und Verzögern von Netzwerkpaketen
- durch Verwendung von anderen Computerprogrammen, die unmittelbar beim Spielen helfen
- durch Ausnutzung eines Bugs
- durch Angriffe auf Netzwerkebene
- durch Absprache u.v.m.

Im Folgenden wird nur auf Cheats, die unmittelbar mit der Zeitsteuerung von Spielen zusammenhängen, eingegangen.

2.5 Zeitbasierte Cheats

Mit zeitbasierten Cheats sind Cheats auf Protokollebene gemeint, bei denen die zeitliche Abfolge von Spielereignissen ausgenutzt oder verändert werden. Dies geschieht meist durch das Abfangen, Ändern, Blockieren oder Verzögern von Nachrichten. Nachfolgend werden einige bekannte Cheats erklärt.

2.5.1 Lookahead Cheat

Wartet ein Spieler mit dem Senden seiner Daten s_i^k für den Frame f_i bis er von allen anderen Spielern die Updates $s_i^{l \neq k}$ für Frame f_i bekommen hat, so kann er basierend auf der ihm nun vorliegenden Information vorausschauend (engl.: *look ahead*) handeln, als letzter sein Update senden und behaupten, er hätte gleichzeitig mit (oder vor) den anderen Spielern gehandelt. Der Verlauf der Nachrichten ist in Abbildung 1 dargestellt.

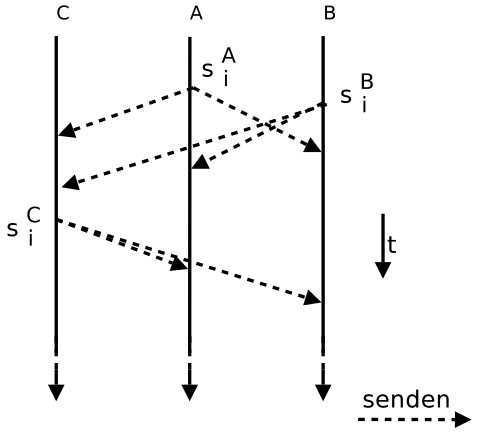

Abbildung 1: *Lookahead Cheat:* Spieler C wartet absichtlich auf die $s_i^{k \neq C}$ der anderen Spieler

2.5.2 Suppress-correct Cheat

Diese Variante bekommt besonders in Verbindung mit Dead-Reckoning (siehe Abschnitt 2.2) Bedeutung. Bei diesem Cheat unterbindet der Spieler p^i für n aufeinander folgende Frames das Senden seines neuen Status $s^i_k \dots s^i_{k+n}$. Dabei darf er n nicht zu groß wählen, da die Verbindung des Spielers sonst als abgebrochen angesehen werden kann. Nun kann er basierend auf den $s^{j\neq i}_k \dots s^{j\neq i}_{k+n}$ eine Aktion ausführen, die Vorteilhaft für ihn ist und dennoch als korrekt anerkannt wird, zum Beispiel ein Ausweichmanöver (siehe Abbildung 2).

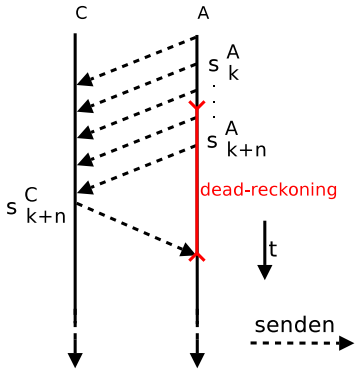

Abbildung 2: *Suppress-correct Cheat:* Spieler C sendet absichtlich keine Updates

2.5.3 Fixed Delay Cheat

Im Unterschied zur gezielten Unterdrückung von Updates, kann ein Spieler auch all seine ausgehenden Pakete um eine feste Zeit verzögern. In Spielen, die dies nicht ausgleichen, kann das einen Vorteil ergeben, da der Cheater alle Updates von anderen Spielern schnell bekommt, die anderen jedoch seine nur langsam.

Eine Übersicht über diese und weitere bekannte Cheats in Onlinespielen sowie über einige der im Folgenden behandelten Lösungsansätze liefert [WeSo07]. Eine Einteilung und Abgrenzung zu klassischen Sicherheitsproblemen findet man in [YaRa05] .

3 Lösungsansätze

3.1 Lockstep

Als grundlegenden Ansatz zur Verhinderung von zeitbasierten Cheats wird ein sogenanntes *stop-and-wait* Protokoll eingeführt. Das bedeutet in diesem Kontext, dass bevor im Spiel die Aktionen für Frame f_{t+1} ausgeführt werden können, die Spieler warten müssen bis sie den Status von allen anderen Spielern für den Frame f_t empfangen haben. Damit ist es nicht mehr möglich sein Update längere Zeit nicht zu senden. Ein Spieler kann aber immer noch warten bis er einige Aktionen von anderen Spielern kennt und dann erst seine Aktion ausführen (Lookahead Cheat). Das *Lockstep* Protokoll aus [BaLL07] verzichtet auf Grund der beschriebenen Probleme (s. Abschnitt 2.2) auf Dead-Reckoning.

Lockstep verhindert den Lookahead Cheat, indem es einen zusätzlichen Schritt einführt. Hat ein Spieler seine Aktion für einen Frame f_i abgeschlossen, sendet er nicht seinen Status s^k_i sondern einen kryptographischen Hash seines neuen Status, $hash(s^k_i)$. Haben nun alle Spieler

ihre Hashs gesendet und die der Anderen empfangen, können alle Spieler ihren Status s_i unverschlüsselt verteilen. Erst danach können die Spieler mit der nächsten Aktion fortfahren.

Mit Hilfe von $hash(s_i)$ eines Spielers kann vor seiner Aktion jeder Spieler feststellen, ob der $hash(s_i)$ des Spielers zu seinem gesendeten s_i passt, indem er den Hashwert von s_i neu berechnet und mit dem empfangenen Hashwert vergleicht. Keiner der Spieler hat nun den Status eines anderen Spielers, bevor er selbst den Hash seines Eigenen gesendet hat. Der Ablauf ist in Abbildung 3 noch einmal verdeutlicht.

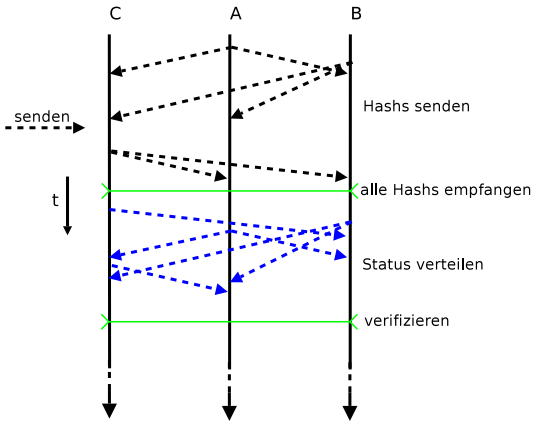

Abbildung 3: *Lockstep*: Spieler warten auf Hashs, bevor sie den neuen Status bekommen und können anschließend prüfen, ob der Status nochmal nachträglich verändert wurde

Das Lockstep Protokoll ist zwar sicher bezüglich zeitbasierter Cheats, schränkt aber auch den Spielfluß ein, denn alle Spieler müssen auf den Status der Anderen warten bevor sie weiter agieren können. Das Spiel ist also immer nur so schnell wie der langsamste Teilnehmer, was sich insbesondere bei ungleichen Netzwerkverbindungen auswirkt. Ein weiterer Nachteil ist, dass zuerst die Hashs verteilt werden und erst anschließend die Updates. Es wird also eine zusätzliche Kommunikationsphase für jeden Frame benötigt. Insgesamt ist das Lockstep Protokoll damit für schnelle Spiele nicht geeignet.

3.2 Asynchronous Synchronization

Mit dem Lockstep Protokoll werden unabhängig von der Situation im Spiel alle Spieler synchronisiert. Dadurch entsteht der sehr hohe zusätliche Aufwand, der oft nicht nötig ist. In [BaLL07] wird Lockstep deshalb zu Asynchronous Synchronization (AS) erweitert, mit dem Ziel die von Lockstep mitgebrachte Cheatsicherheit auszunutzen, aber nur dann zu verwenden wenn dies auch notwendig ist.

Die Idee ist, dass jeder Spieler asynchron seine Frames spielen kann, bis zu einem Zeitpunkt an dem ein Konflikt mit einem anderen Spieler auftreten kann. Dazu muss ständig verfolgt werden in welchem Bereich eines Spielers p^i alle anderen Spieler p^j im Frame f_k^i eingreifen könnten. Dieser Einflussbereich (EB) ist abhängig von der Art des Spiels und sei hier vereinfacht ein Kreis. Realistischer wären zum Beispiel Kugeln oder Sichtkegel. Sein Radius wird anfangs so gewählt, dass er die maximale Veränderung des EBs innerhalb eines Frames umschließt.

Als erstes führt nun Spieler p^i seine Aktion aus und verteilt den Wert $hash(s_k^i)$. Danach schaut er sich $s_{l\leq k}^{j\neq i}$ aller anderen Spieler p^j nacheinander an:

1. Spieler p^i berechnet seinen EB^i

2. p^j berechnet EB^j vom letzten bekannten s^j_{last}, indem der EB in Korrespondenz zum Frameunterschied $k - last$ skaliert wird
3.

EB schneiden sich:	EB schneiden sich nicht:
warten bis nächstes s^l bekannt, um neuen EB auszurechnen	warten ist nicht nötig, sende s^i_k weiter zu f^i_{k+1}

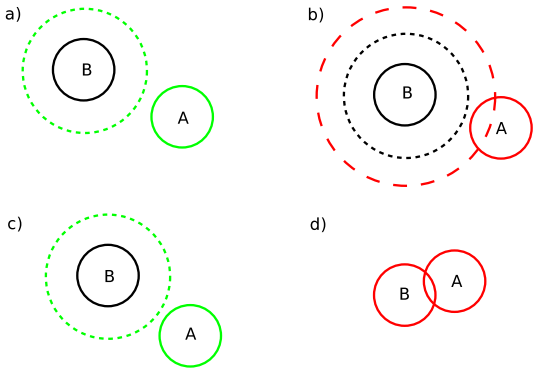

Abbildung 4: Berechnung von Einflussbereichen bei AS

In Abbildung 4 a) liegt aus der Sicht des Spielers A die letzte bekannte Position von B einen Frame zurück. Die EB schneiden sich nicht, also kann A unabhängig von B den nächsten Frame ausführen. In b) hat Spieler A keine neue Position von B erhalten. Die errechneten EB schneiden sich, also muss A mit der nächsten Aktion warten bis eine neuere Position von B bekannt ist. In c) hat A einen neueren Status von B bekommen, in dem die EB sich nicht schneiden. Damit kann B asynchron weiter fortfahren. Zu einem späteren Zeitpunkt in d) kommt es zur Interaktion, es wird wie mit Lockstep verfahren.

Die Spieler können damit, solange sie sich nicht gegenseitig beeinflussen können, unabhängig voneinander im Spielverlauf fortschreiten. Tritt allerdings eine Konfliktsituation auf, dann muss gewartet werden bis der Konflikt sich aufgelöst hat bzw. interagiert wurde. Je nach Situation muss der schnellere Client dann nur einen Frame warten, weil die neu errechneten EB sich dann schon nicht mehr schneiden. Im schlechtesten Fall muss allerdings so lange gewartet werden, bis der weiter hinten liegende Spieler den schneller Spieler eingeholt hat, um dann gemäß Lockstep zu interagieren.

Bei der Verwendung von AS muss für das Spiel irrelevant sein in welchem groben Bereich ein anderer Spieler sich aufhält, solange er außerhalb seines EB ist. Ein Spieler kann keinen Lookahead Cheat anwenden, da bei möglichen Konflikten immer Lockstep angewendet wird und er zwar weiß, wo sich ungefähr sein Kontrahent aufhält, aber ihn nicht beeinflussen kann.

Eine Möglichkeit das Wissen der ungefähren Position anderer Spieler zu verschleiern bieten zellbasierte Protokolle, die nur dann einen genauen EB berechnen, wenn zwei Spieler in nahen Zellen sind. Eine Erweiterung von AS für heterogene Netze liefert [JoLe05].

3.3 Sliding Pipline

Mit AS und Lockstep wurden zwei Protokolle vorgestellt die wegen der einhergehenden Schwierigkeiten komplett auf Dead-Reckoning verzichtet haben. In sehr schnellen Spielen bei denen kurze Latenzen für flüssiges Spielen notwendig sind, ist Dead-Reckoning aber eine bewährte Methode, um die in Netzwerken auftretenden Latenzen zu verstecken. In [FiJa03] wird eine Variante von Lockstep veranschaulicht, die mit Dead-Reckoning zurechtkommt.

Man erweitert das Lockstep Protokoll um eine Pipeline mit n Stufen und verwendet eine Synchronisationsverzögerung. Das bedeutet, dass Aktionen die ein Spieler ausführt nicht sofort übernommen werden sondern erst nach einer fest gegebenen Zeitspanne. Damit können Aktionen die von unterschiedlichen Clients zur gleichen Wallclock Time ausgeführt werden, zur gleichen Zeit nach der Übertragung umgesetzt werden.

Einem Spieler p^i ist es erst erlaubt seinen Status s^i_k zu senden, wenn er von allen anderen Spielern $p^{j\neq i}$ den Hash $hash(s^j_k)$ empfangen hat. Er kann jedoch unter Ausnutzung der Synchronisationsverzögerung bis zu n Hashs mehr senden, als er empfangen hat, also bis zu $hash(s^i_{k+n})$ sofern er alle $hash(s^{j\neq i}_k)$ empfangen hat. Ist dieser Punkt erreicht ist die Pipeline voll und er muss auf die $hash(s^{j\neq i}_{k+1})$ warten.

Anstatt nun auf ein n festgelegt zu sein, verwendet das *Adaptive Pipeline Protocol* (AP) eine dynamische Pipelinelänge. Jeder Spieler p^i errechnet sich aus den eingegangen Hashs die $d^i_{j\neq i}$ und findet so $d^i_{max} := max\{d^i_{j\neq i}\}$. Mit jedem Hash wird nun zusätzlich d^i_{max} mitgesendet, so dass jeder Spieler nun d_{max} kennt. Für jeden Frame f_i wird dann wie folgt vorgegangen:

1. Berechne $n_i = \lceil d_{max} * r_{max} \rceil$ und $\alpha = n_i - n_{i-1}$
2.

$\alpha > 0$	$\alpha < 0$	$\alpha = 0$
sende zusätzlich α Hashs	sende α weniger Hashs	sende genauso viele wie bei f_{i-1}

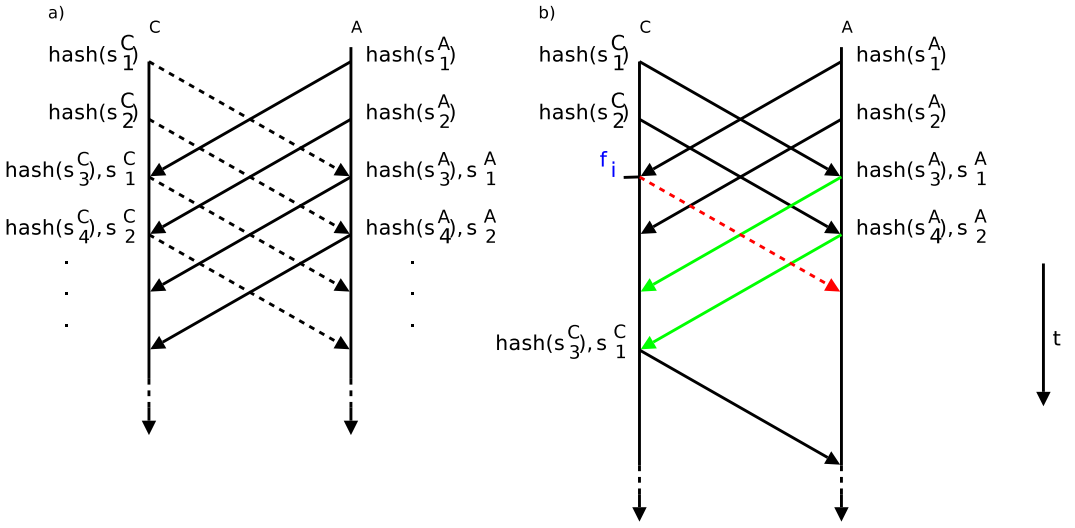

Abbildung 5: *a)* Lockstep mit Pipeline $n = 2$ *b)* late-commit Cheat von Spieler C

Abbildung 5 a) zeigt den Abblauf mit einer Pipelinelänge von $n = 2$. Durch das Pipelining entsteht die Möglichkeit den in Abbildung 5 *b)* mit $n = 2$ veranschaulichten Late-commit Cheat anzuwenden. Spieler p^C erarbeitet sich einen Vorteil, indem er zum Zeitpunkt f_i nicht wie Spieler p^A erwartet, $hash(s^C_3), s^C_1$ sendet. Anstatt dessen wartet er solange, bis er von p^A alle möglichen n Hashs mit den entsprechenden s^A empfangen hat. Nun kann p^C seine nächste Aktion basierend auf $n * s^A$ treffen, während p^A seine Entscheidungen $2 * n$ Frames gegenüber s^C im Voraus treffen muss.

Diesem Cheat kann dadurch begegnet werden, dass p^A die Kenntnis über d^A_C ausnutzt und sich merkt wann s^C_1 ankommt. Ist $|s^C_1 - s^A_1| < d^A_C + \frac{1}{r^2}$ hat p^C sein s^C_1 vor der Ankunft von s^A_1 abgeschickt und cheatet damit nicht. Dadurch kann aber nicht ausgeschlossen werden, das ein Spieler der auf einmal höhere Latenzen hat, fälschlicherweise als Cheater erkannt wird. Insbesondere ist es für diese Vorgehensweise nicht notwendig eines der Pipeline Protokolle zu verwenden. Hier werden die Hashs zur Verhinderung des Cheats nicht ausgenutzt. Eine sichere Erkennung ist nicht möglich, was eine Schwachstelle des Protokolls darstellt.

Im vorgestellten AP Protokoll wird die Länge der Pipeline dynamisch angepasst, wodurch es zu Jitter kommen kann. *Sliding Pipeline* verhindert dies durch die Hinzunahme eines Sendepuffers. Kann nun ein Spieler schon einen Hash senden, das Protokoll ihn aber nicht annehmen, wird der Hash an das Ende des Puffers gehängt. Liegt ein Update noch nicht vor, so wird Dead-Reckoning verwendet, um den Spielfluß zu gewährleisten.

Durch das adaptive Verändern der Pipeline wird der Einfluss von Latenzen auf den Spielfluss minimiert. Ist der Sendepuffer ausreichend groß kann Jitter damit verhindert werden. Der Speed-up gegenüber Lockstep ist proportional zu der Pipelinelänge n. Da es nicht zulässig ist (bzw. erkannt werden kann) Updates zu unterdrücken, ist auch der bei Dead-Reckoning verwendete Suppress-correct Cheat nicht relevant.

3.4 New Event Ordering

Das New Event Ordering Protokoll (*NEO*) aus [GZLM04] spezialisiert sich darauf, eine Cheatsichere Peer-to-Peer Architektur zu erreichen. Es versucht den Spielfluss für die Mehrheit der Spieler und nicht zwingend für alle Spieler zu erhalten.

Der Spielverlauf wird in Runden unterteilt, so dass jeder Spieler innerhalb dieses festgelegten maximalen Zeitintervalls sein Update senden muss. Schafft er dies nicht, weil seine Verbindung zu den anderen Spielern nicht schnell genug für das gewählte Zeitintervall ist, kann er nicht am Spiel teilnehmen.

Das Update wird im Gegensatz zu den anderen hier vorgestellten Protokollen nur verschlüsselt übertragen. Der zugehörige Schlüssel wird in der nächsten Runde mitgesendet. In Abbildung 6 wird der Aufbau einer Nachricht dargestellt. Es wird der signierte und verschlüsselte Status, der Schlüssel um den Status der letzten Runde zu entschlüsseln und noch ein Bit-Vektor gesendet. In Peer-to-Peer Architekturen kann es vorkommen, dass p^i das verschlüsselte s_l^k innerhalb der Rundenzeit empfangen hat, p^j aber nicht. Um Inkonsistenzen zu vermeiden wird ein Konsens gefunden, wann ein s_l^k eines Spielers akzeptiert wird und wann nicht. In NEO wird ein s_l^k akzeptiert, wenn die Mehrheit der Spieler dieses empfangen hat. Im Bitvektor am Ende der Nachricht gibt p^i an, ob er die $s^{i \neq j}$ empfangen hat. Jeder Spieler bekommt zum Zeitpunkt f_l eine Übersicht von allen anderen Spielern, ob sie s_{l-1} empfangen haben und von welchen Spielern. Ist dann immer noch unklar, ob ein s_{l-1} akzeptiert wurde, muss bei einem anderen Spieler, dessen Nachricht nicht angekommen ist, nachgefragt werden.

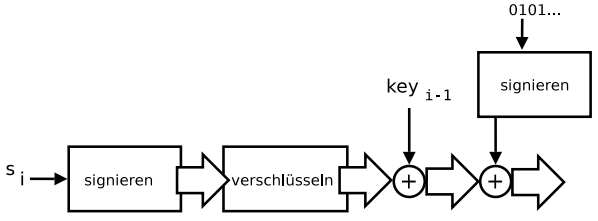

Abbildung 6: Format einer Nachricht bei NEO

Kommt ein Status s_{l-1}^i eines Spielers p^i nicht bei p^k an, kann ohne Unterbrechung weitergespielt werden, wenn dies bei der Mehrheit der anderen Spieler auch der Fall war. Für den Spieler dessen s_{l-1}^i nicht bei der Mehrheit angekommen ist, wird Dead-Reckoning verwendet. Der Spieler p^i muss seinen Status gegebenenfalls nachbessern. Es ist damit aber sichergestellt, dass die Mehrheit der Spieler davon nicht betroffen ist.

Um die Performanz von NEO zu erhöhen wird, ähnlich wie beim Piplined Lockstep Protokoll (s. Abschnitt 3.3), eine Pipeline verwendet. Die Tiefe der Pipeline wird dabei festgelegt auf

$n = \frac{Rundenzeit}{t}$, wobei t hier die Zeit zwischen Rundenstart und Ankunft des nächsten Updates meint (siehe Abbildung 7). Ferner wird nun nicht mehr der Schlüssel der letzten Nachricht mitgesendet sondern der von s_{i-n}.

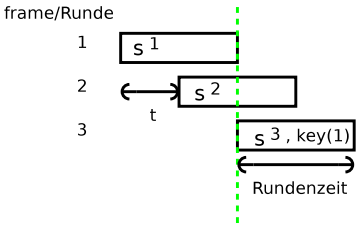

Abbildung 7: Pipeline bei NEO

Mit dem angehängten Bitvektor kann noch mehr kodiert werden. Es kann damit etwa die Rundenlänge oder die Frequenz der Updates dynamisch angepasst werden. Ein Spieler kann zudem die Frequenz der Updates direkt an die des Spielers an den er sie sendet anpassen, damit wird der Suppress-correct Cheat vernachlässigbar. NEO erlaubt dank der festgelegten Rundenzeit in der ein Update gesendet werden muss keine verspäteten Updates und verhindert so insgesamt zeitbasierte Cheats (und darüber hinaus auch einige andere Cheats).

Auf Grund der Art wie in NEO die Nachricht zusammengesetzt ist, ist es anfällig für einige Attacken auf Netzwerkebene. Mit dem *Secure Event Agreement* werden diese behoben. Hierbei wird die Nachricht um eine Session ID und eine Absender ID erweitert und diese nicht verschlüsselt, sondern mittels eines kryptographischen Hashes verifiziert. Die Details finden sich in [CDST06].

3.5 AC/DC

Bei den bisherigen Ansätzen wurde mit Hilfe von Protokollen versucht Cheating von vorne herein auszuschließen. Einen anderen Ansatz verfolgt der *Algorithm for Cheating Detection by Cheating* (*AC/DC*) , indem versucht wird in Peer-to-Peer Architekturen Cheater zu erkennen. Anschließend können entsprechende Maßnahmen eingeleiten werden (z.B.: Ausschluss eines Spielers aus dem Spiel).

AC/DC stützt sich auf ein allgemeineres Modell des Spielverlaufs basierend auf der Wallclock Time und der Zeit aus der Sicht des Spielers. Der Algorithmus wird in [FeRo06a] beschrieben und konzentriert sich auf die Erkennung des Lookahead Cheats. Zuerst muss ein Spieler in Verdacht geraten zu cheaten. Ein Spieler p^i kann beispielsweise verdächtigt werden, wenn seine Verzögerungen $d^i_{j \neq i}$ zu einem oder mehreren Spielern p^j einen ungewöhnlichen Verlauf haben, zum Beispiel eine konstante lineare Veränderung. Oder wenn bisher beobachtet wurde, dass $d^i_j > d^i_k \wedge d^i_j < d^i_l$ gilt, aufeinmal aber $d^i_j > d^i_k \wedge d^i_j > d^i_l$ ist. Schließlich kann ein Spieler natürlich auch in Verdacht geraten, wenn er sehr gute oder ungewöhnlich gute Ergebnisse im Spiel erzielt.

Wird ein Spieler p^c von p^a verdächtig, versucht AC/DC zu erkennen, ob tatsächlich ein Lookahead Cheat angewendet wird. Wendet p^c den Cheat an, dann wartet er auf ein s^a_k, berechnet daraus ein für sich selbst vorteilhaftes s^c_k. Diesen Status verteilt er dann und behauptet, es sei gleichzeitig zu dem Empfangenen entstanden. Der Spieler p^a misst nun für eine bestimmte Zeit d^a_c, um einen Mittelwert γ als Vergleichswert zu haben. Anschließend schickt p^a seine s^a in Richtung p^c um δ für eine gewisse Zeit verzögert und misst dabei erneut d^a_c, um den neuen Mittelwert γ^{neu} zu berechnen. Ist nun

$$\gamma^{neu} > \gamma + \alpha,$$

wobei α eine sinnvoll gewählte Konstante ist, die einen deutlichen Unterschied zwischen zwei gemittelten Zeitwerten kennzeichnet, dann bestätigt p^a den Verdacht. Denn p^c hat auf die künstlich eingeführte Verzögerung δ reagiert. Ist dies nicht der Fall, dann wird δ erhöht und der Vorgang wiederholt sich.

Mit der Inkrementierung von δ wird solange fortgefahren bis mit γ^{neu} ein Wert Γ erreicht wird. Danach gilt der Verdacht als nicht bestätigt und das Spiel kann mit zurückgesetztem δ fortgesetzt werden. Γ ist ein zusammengesetzter Wert, eine obere Schranke, die größer ist als $max\{d_c^k, p^k \in \mathcal{P}\}$, zusätzlich den maximalen Unterschied im Spielfortschritt zwischen je zwei Spielern und dass p^c auch auf $s^{k \neq a}$ von anderen Spielern warten könnte, berücksichtigt.

Die Vorgehensweise von AC/DC setzt voraus, dass ein Spieler über Konfliktsituationen entscheiden kann und die Cheaterkennung kontrolliert, was das Problem der Auswahl eines dafür geeigneten Spielers aufwirft. Möglich wäre eine Weitergabe dieses Privilegs, was Gegenmaßnahmen von Cheatern erschweren würde. Denn ein Cheater könnte merken, dass er unter Verdacht geraten ist und das Cheating für eine gewisse Zeit unterlassen. Wird ein Spieler fälschlicherweise als Cheater verdächtigt, entsteht für ihn für eine kurze Zeit ein Nachteil auf Grund der durch die Detektion eingeführten Verzögerung. Abgesehen davon entstehen keine Einschränkung der Performanz. Es ist allerdings auch nicht sichergestellt, dass jeder Cheater erkannt wird, da er verhindern könnte in Verdacht zu geraten.

In [FeRo06b] wird die Methode zu Detektion auf weiterer zeitbasierte Cheats erweitert.

3.6 RACS

Mit dem Referee Anti-Cheat Scheme wird in [WeSL07] eine Architektur vorgestellt, die sowohl die Vorteile der Client-Server als auch der Peer-to-Peer Architektur ausnutzt, die effizient ist und trotzdem zeitbasierte Cheats verhindert.

Dazu benötigt man einen vertrauenswürdigen Host, den Referee, der folgende Aufgaben hat:

- Verifizierung und Konfliktauflösung
- Verteilen der Updates unter den Spielern
- Den momentanen Spielstatus verwalten

Zur Verwaltung des Spielstatus wird der Spielverlauf wieder in konkrete Zeitschritte unterteilt. Zu Spielbeginn initialisiert der Referee f_0 und synchronisiert die Spieler. Jede Nachricht von p_k^i beinhaltet k und die Aktionen a_k^i, nicht aber die kritische Informationen, wie der Zustand des Spielers (also $a_k^i \subset s_k^i$). Es gibt nun drei verschiedene Arten von Nachrichten.

1. $MPP(k, a_k^i)$ - Nachricht direkt von Spieler zu Spieler (Peer-to-Peer), beinhaltet nur die Informationen über die eigenen Aktionen, die in allen Nachrichten mitgesendet werden
2. $MPR(k, a_k^i, s_k^i, b_k^{j \neq i})$ - Nachricht vom Spieler zum Referee (Peer-to-Referee), hier wird der gesamte Status von p^i und zusätzlich mit $b_k^{j \neq i}$ die Hashs, der von p^i direkt empfangenen Updates, mit denen der Referee die Integrität prüfen kann, übertragen
3. $MRP(k, a_k^i, i)$ - Nachricht vom Referee zum Spieler p^i (Referee-to-Peer),

Zur Verifikation der Nachrichten wird Public-Key Kryptographie verwendet.

In RACS wird zwischen 2 Modi unterschieden: Im PP Modus werden zusätzlich die MPP wie beschrieben verwendet, um die notwendigen Daten schnell zwischen den Spielern auszutauschen. Im PRP Modus findet keine direkt Kommunikation zwischen den Spielern statt, alles wird über den Referee geroutet.

Anfangs befinden sich die Spieler im PRP Modus. Schneiden sich ihre Einflussbereiche, initiiert der Referee den PP Modus. Ein Spieler p^i ordnet an aus dem PP Modus mit Spieler p^k wieder zurück auf den PRP Modus zu kehren, wenn gilt:

1. p^k und p^i Einflussbereiche schneiden sich nicht mehr
2. mehr als ein festgelegter Prozentsatz c_1 an Nachrichten geht verloren
3. p^i erhält für eine bestimmte Anzahl c_2 von Frames keine Updates mehr von p^k

Mit den beiden Konstanten c_1, c_2 kann ein gewisser Trade-Off zwischen Cheatsicherheit und Performanz eingestellt werden. Wobei mit $c1 = 0\%$ und $c2 = 1$ die cheatsicherste Einstellung gewählt ist.

Kommt eine Nachricht für f_k^i nicht beim Referee an, dann verwendet dieser Dead-Reckoning und benachrichtigt alle betroffenen Spieler $p_k^{j \neq i} \cup p_k^i$ über den neuen Status und überschreibt damit eventuell bei den Spielern angekommene Nachrichten. Es können somit keine Updates zurückgehalten werden, der Suppress-update Cheat ist also nicht möglich.

Kommt ein Status zu spät für eine Runde an, wird er für die folgende Runde verwendet, sofern dafür keine Nachrichten existieren. Ist dies der Fall, wird die Nachricht verworfen, was den Lookahead Cheat verhindern kann, wenn die Rundenzeit nicht zu groß ist.

Wendet ein Spieler den Fixed-delay Cheat an, kann für den Cheater eine höhere Verzögerung dadurch entstehen, dass die Nachrichten nicht mehr rechtzeitig ankommen und wieder auf den PRP Modus geschaltet wird. Ist die Verzögerung allerdings nur gering so behandelt RACS dies nicht als Cheat, da sonst Spieler mit schwachen Verbindungen ausgeschlossen werden.

4 Zusammenfassung

Mit dem Lockstep Protokoll wurde ein bezüglich zeitbasierter Cheats sicheres Protokoll vorgestellt, dass für die Praxis auf Grund des erhöhten Aufwands allerdings nur begrenzt geeignet scheint. Die Erweiterung zu Asynchronous Synchronization bietet Spielern die sich unabhängig voneinander bewegen die Möglichkeit asynchron im Spielverlauf voranzukommen, kommt aber im schlechtesten Fall nicht an der langsamen Lockstepsynchronisation vorbei. Die höhere Performanz von Sliding Pipeline erkauft man sich durch den Nachteil, dass es einen Art von Cheat nur unsicher erkennen kann. Eine ganz andere Vorgehensweise setzt das New Event Ordering Protokoll ein, dass es der Mehrheit der Spieler ermöglicht einen stabilen Spielfluß zu haben. AC/DC ermöglicht eine Erkennung von Cheatern, wenn diese unter Verdacht geraten und verhindert so eine ständige Einschränkung des Spielflusses. Allerdings ist es anfällig für Gegenmaßnahmen. Schließlich wurde mit RACS eine sich an die Gegebenheiten anpassende Architektur vorgestellt, die bei Verdacht auf Cheating den schnelleren Peer-to-Peer Modus zurückfährt und den Referee die Kommunikation kontrollieren lässt.

Von entscheidender Bedeutung welche Art von Verfahren letztendlich Verwendung findet, ist sicherlich das Genre des Spiels. So haben MMORPGs ganz andere Anforderungen an Ressourcen und Geschwindigkeit als FPS.

Die hier vorgestellten zeitbasierten Cheats bilden nur eine kleine Untermenge der möglichen Cheats in Onlinespielen und können bereits zu einigen Komplikationen führen. Das zeigt, dass hier noch Spielraum für Ideen vorhanden ist. Vorallem weil noch nicht viel auf diesem Bereich der Sicherheit veröffentlicht worden ist.

Literatur

[BaLL07] Nathaniel E. Baughman, Marc Liberatore und Brian Neil Levine. Cheat-proof playout for centralized and peer-to-peer gaming. *IEEE/ACM Trans. Netw.* 15(1), 2007, S. 1–13.

[CDST06] Amy Beth Corman, Scott Douglas, Peter Schachte und Vanessa Teague. A Secure Event Agreement (SEA) protocol for peer-to-peer games. *Availability, Reliability and Security, International Conference on* Band 0, 2006, S. 34–41.

[FeRo06a] Stefano Ferretti und Marco Roccetti. AC/DC: an algorithm for cheating detection by cheating. In *NOSSDAV '06: Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*, New York, NY, USA, 2006. ACM, S. 1–6.

[FeRo06b] Stefano Ferretti und Marco Roccetti. Game time modelling for cheating detection in P2P MOGs: a case study with a fast rate cheat. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, New York, NY, USA, 2006. ACM, S. 35.

[FiJa03] Eric Cronin Burton Filstrup und Sugih Jamin. Cheat-Proofing Dead Reckoned Multiplayer Games (Extended Abstract). In *Proc. of 2nd International Conference on Application and Development of Computer Games*, 2003.

[GZLM04] Chris GauthierDickey, Daniel Zappala, Virginia Lo und James Marr. Low latency and cheat-proof event ordering for peer-to-peer games. In *NOSSDAV '04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, New York, NY, USA, 2004. ACM, S. 134–139.

[JoLe05] Aaron St. John und Brian Neil Levine. Supporting P2P gaming when players have heterogeneous resources. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, New York, NY, USA, 2005. ACM, S. 1–6.

[WeSL07] Steven Daniel Webb, Sieteng Soh und William Lau. RACS: A Referee Anti-Cheat Scheme for P2P Gaming. In *17th International Workshop on Network and Operating Systems Support for Digital Audio and Video June 4-5*, 2007, S. 34–42.

[WeSo07] Steven Daniel Webb und Sieteng Soh. Cheating in networked computer games: a review. In *DIMEA '07: Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, New York, NY, USA, 2007. ACM, S. 105–112.

[YaRa05] Jeff Yan und Brian Randell. A systematic classification of cheating in online games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, New York, NY, USA, 2005. ACM, S. 1–9.

Abbildungsverzeichnis

Sicherheit von Routingprotokollen

Fabian Geisberger

Das Internet ist ein Zusammenschluss aus vielen einzelnen Netzwerken. Zwischen diesen Netzwerken wird das Border Gateway Protocol (BGP) als dynamisches Routingprotokoll eingesetzt. Zuerst wird die Funktionsweise von BGP beschrieben. Danach wird beschrieben, wie die aktuelle Struktur des Internets ausgenutzt werden kann, um Traffic gezielt umzulenken. Anschließend werden zwei Erweiterungen für BGP, das Secure Border Gateway Protocol (S-BGP) und das Secure Origin Border Gateway Protocol (soBGP), sowie deren Funktionsweise vorgestellt, die diese Probleme lösen sollen. Zudem werden allgemeine Sicherheitsvoraussetzungen an Routingprotokolle in Form von rpsec erläutert. Zum Schluss wird noch der YouTube-Pakistan-Zwischenfall als populäres Beispiel für Probleme des aktuellen Protokolls vorgestellt.

1 Einleitung

Das Internet unterliegt einer ständigen Veränderung. Neue Verbindungen zwischen Netzen werden aufgebaut, bestehende Verbindungen werden gewollt oder ungewollt getrennt. Das Netz muss sich diesen geänderten Gegebenheiten selbstständig anpassen. Zu diesem Zweck werden dynamische Routingprotokolle eingesetzt. Im Internet hat sich hier BGP (Border Gateway Protocol) durchgesetzt. Doch beim Design des Protokolls stand mehr die Funktion als die Sicherheit im Vordergrund.

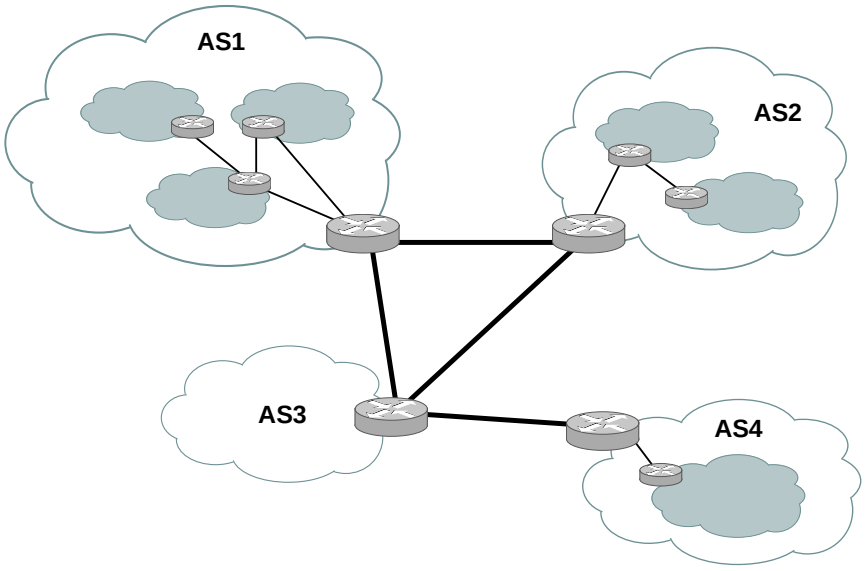

Abbildung 1: Schematischer Aufbau des Internets

1.1 Autonome Systeme

Um den Aufbau des Internet zu verstehen, soll zunächst der Begriff des Autonomen Systems geklärt werden. Wie in Abb. 1 symbolisch dargestellt, besteht das Internet aus vielen unabhängigen Systemen, sog. Autonomen Systemen (AS). Ein AS ist ein Netzwerk bzw. eine Gruppe von Netzwerken, die zentral verwaltet und administriert werden. Die interne Struktur eines AS ist dabei für den Außenstehenden irrelevant. Jedes AS wird dabei durch eine Nummer repräsentiert [Beij02]. Beispiele für Autonome Systeme sind: AS553 – BELWUE (Landeshochschulnetz Baden-Württemberg) und AS3320 – DTAG (Deutsche Telekom AG).

1.2 Routingprotokolle

Nicht nur zwischen Autonomen Systemen werden Routingprotkollen eingesetzt, sondern auch innerhalb eines AS. Man gliedert die Routingprotokolle dabei in zwei Klassen. Folgende Übersicht stellt die bekanntesten Routing-Protokolle dar: [Beij02]

- Interior Gateway Protocols
  - RIP (Routing Information Protocol)
  - OSPF (Open Shortest Path First)
  - IGRP (Interior Gateway Routing Protocol)
  - EIGRP (Enhanced Interior Gateway Routing Protocol)
  - IS-IS (Intermediate System to Intermediate System)
- Exterior Gateway Protocols
  - EGP (Exterior Gateway Protocol)
  - BGP (Border Gateway Protocol)

Da ein AS normalerweise eine zentrale Administration hat, ist die Sicherheit innerhalb des AS viel leichter zu gewährleisten. Neue Sicherheitsmechanismen können direkt bei allen Geräten eingetragen werden. Im Internet hingegen werden die Grenzrouter eines AS vom jeweiligen AS betreut und unterliegen keiner zentralen Administration. Diese Arbeit setzt ihren Schwerpunkt daher auf die Sicherheit im Inter-AS-Routing.

2 BGP

BGP wurde erstmals 1989 als RFC 1105[1] als Ablösung des damals eingesetzten EGP (Exterior Gateway Protokoll) spezifiziert. Seit dem wurde es ständig weiter entwickelt. Die aktuelle Version ist BGPv4 (RFC 4271[2]) vom Januar 2006. Wenn heutzutage von BGP gesprochen wird, ist fast ausschließlich BGPv4 gemeint.

2.1 Funktionsweise

Zwei Router tauschen über den TCP Port 179 Nachrichten aus. Diese Nachrichten dienen dazu, neue Routen bekannt zu geben und alte Routen zu widerrufen.

Erhält dabei ein Router eine neue Route von seinem Gegenüber, werden folgende Schritte abgearbeitet: [Beij02]

[1] http://tools.ietf.org/html/rfc1105
[2] http://tools.ietf.org/html/rfc4271

1. Es wird geprüft, ob die eingehenden Filter für die aktuelle BGP-Session die Route erlauben, falls nicht, wird diese verworfen.
2. Die Route wird in die BGP-Tabelle eingefügt.
3. Auf Basis der neuen Tabelle wird für den aktuellen Netzbereich die neue beste Route ermittelt. Falls die neue Route nicht die beste Route ist, stoppt die Verarbeitung.
4. Die alte beste Route wird durch die neue in der Routing-Tabelle ersetzt.
5. Die alte beste Route wird bei allen BGP-Nachbarn, welche diese Route erhalten haben, widerrufen.
6. Die neue beste Route wird, falls nicht durch Filter verhindert, an alle BGP-Nachbarn bekannt gegeben.

Bei Schritt 3 wird die beste Route entweder durch eingetragene Präferenzen oder durch den kürzesten AS-Pfad gewählt. Es spielt dabei keine Rolle, wie lange der wirkliche Pfad durch das AS ist.

2.2 Verwundbarkeit

Die Sicherheit von BGP basiert einzig und allein auf der Annahme, dass sämtliche BGP-Router korrekt arbeiten. Korrekt bedeutet dabei folgendes: [KeLS00]

- Jede BGP-Nachricht stammt vom angegeben Absender und wurde auf dem Weg vom Absender zum Empfänger nicht verändert.
- Jede BGP-Nachricht erreicht nur den Empfänger, für den sie bestimmt war.
- Der Absender der BGP-Nachricht ist von seinem AS berechtigt, die Route an das AS des Empfängers weiterzuleiten.
- Der Besitzer des Adressraums der Route ist auch wirklich der eingetragene Eigentümer (z.B. beim RIPE)
- Das erste AS in der Route war berechtigt, den Adressraum für sein AS bekannt zu geben.
- Falls die Nachricht einen Widerruf enthält, dann war der Widerrufende vorher dazu berechtigt, die Route bekannt zu geben.
- Der Absender der BGP-Nachricht hat die für ihn geltenden BGP-Richtlinien sowie die AS-Richtlinien im Bezug auf Routenauswahl und Routenweitergabe korrekt angewandt.
- Der Empfänger der BGP-Nachricht wendet die für ihn geltenden BGP-Richtlinien sowie die AS-Richtlinien im Bezug auf die Akzeptierung von neuen Routen korrekt an.

Sicherheitsrisiken können dabei entweder durch die Implementierung von BGP oder durch Designprobleme entstehen. Implementierungsbasierte Risiken entstehen dabei z.B. bei der Umsetzung der BGP-Spezifikation in die Software und betreffen nur die jeweilige Software. Designprobleme hingegen betreffen sämtliche Implementierungen, daher beschränken wir uns auf diese.

Ein Angriff auf BGP kann ganz klassisch durch Mitlauschen auf der Leitung zwischen zwei BGP-Nachbarn oder durch das unautorisierte Eindringen in den Router und Veränderung der dortigen Software erfolgen. Verstärkte Zugangskontrolle zu den Netzwerkräumen und Routern sowie eine Verschlüsselung zwischen den BGP-Nachbarn begrenzt diese Angriffsfläche. Diese Gegenmaßnahmen bieten aber keinen Schutz gegen absichtliche oder versehentliche Fehlkonfiguration durch den Administrator eines Routers. [Kent03]

3 Traffic umleiten leicht gemacht

Wie die Analyse von BGP gezeigt hat, lässt sich der Trafficstrom eines kompletten Adressraumes beinträchtigen. Dabei sollten allerdings folgende Grundlagen beachtet werden: [PiKa08]

- Um Traffic weiterzuleiten, benötigt man die Kontrolle über ein eigenes oder fremdes AS.
- Auch wenn die Trafficumleitung auf den ersten Blick nicht auffällt, so sieht man sie, wenn man danach sucht.
- Man ist nicht unsichtbar. Damit die Umleitung funktioniert, müssen die fremden Internetrouter das eigene AS kennen. Damit kann der Übeltäter im Normalfall relativ schnell lokalisiert werden.
- Man braucht BGP Nachbarn, die die eigenen gefälschten Nachrichten auch wirklich weiterleiten und sie nicht auf Grund von lokalen Richtlinien verwerfen.

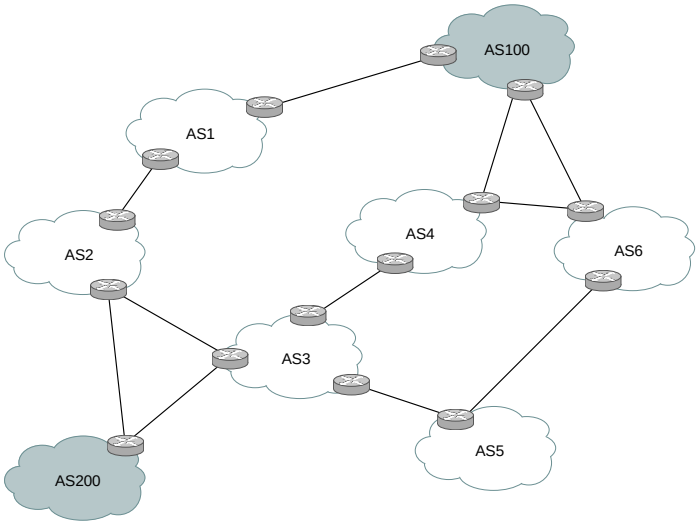

Abbildung 2: Struktur für den Test-Aufbau

3.1 Traffic umleiten

Sind nun all diese Grundüberlegungen angestellt, kann mit dem Angriff begonnen werden. Als Beispiel sei die Struktur wie in Abb. 2 gegeben. Wir möchten dabei den Traffic von AS200, genauer 1.2.3.0/24, umleiten und kontrollieren selbst das AS100. AS200 übermittelt seine Routen an AS2 und AS3, welche sie weiterschicken, bis sie auch bei uns im AS100 angekommen sind. Als Ergebnis entsteht ein Routing wie in Abb. 3 durch die gestrichelten Pfeile dargestellt.

Aus unserer Sicht ist also das Netz 1.2.3.0/24 über den Pfad AS1, AS2 und AS200 erreichbar. Wir geben nun die Route für 1.2.3.0/24 bekannt und behaupten dabei direkt mit AS1, AS2 und AS200 verbunden zu sein. Daraufhin ändert das Routing wie in Abb. 4 dargestellt.

Wir müssen nun nur noch eine statische Route für 1.2.3.0/24 über unsere BGP-Nachbarn bei AS1 eintragen. Jetzt schicken AS3, AS4, AS5 und AS6 ihren kompletten Traffic für AS200 an uns. Dieser kann beliebig analysiert oder verändert werden und wird abschließend über AS1 zum Empfänger weitergeleitet. [PiKa08] Treten keine Fehler auf, erfolgt die Änderung der Routen völlig transparent und ohne dass dies von einem Außenstehenden bemerkt wird. Dafür ist schließlich BGP konzipiert worden.

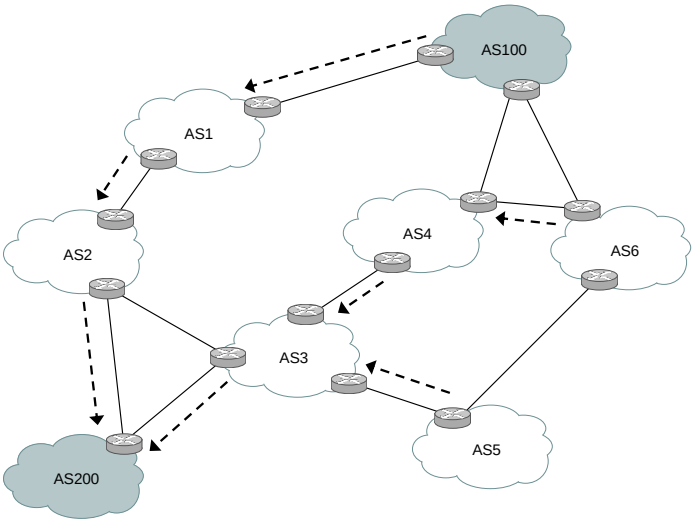

Abbildung 3: Routing für 1.2.3.0/24

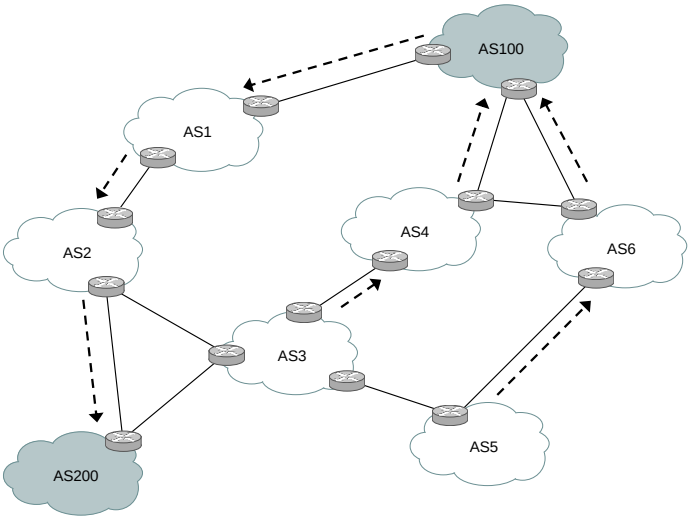

Abbildung 4: Routing für 1.2.3.0/24 nach dem bekanntgeben unserer gefälschten Route.

3.2 Abschließendes

Würde man nun ein Traceroute von z.B. AS3 nach 1.2.3.0/24 durchführen, so würde dieser die Zwischenpunkte von AS4, AS100, AS1, AS2 und AS200 aufweisen. Beim normalen Trafficfluss wäre es nur AS3 und AS200. Um diesen Schönheitsfehler zu beheben, kann in AS100 die TTL[3] der durchgeleiteten Pakete verändert werden. Dadurch wird vorgegaukelt, das Paket hätte deutlich weniger Zwischenpunkte passiert, als das in Wirklichkeit der Fall ist. Im AS-Pfad der Router taucht der Angreifer trotzdem auf. [PiKa08]

[3]Time-To-Live

4 S-BGP

Bereits 1996 begann das Internetwork Research Department von BBN Technologies die Erweiterung S-BGP (Secure BGP) für BGPv4 zu entwickeln, welche BGP um die fehlenden Sicherheitsmerkmale erweitern sollte.

4.1 Aufbau

S-BGP besteht dabei aus 4 Teilen: [Kent03]

- Einer *Public Key Infrastructure (PKI)*, welche die Eigentümer von Adressraum und AS-Nummern abbildet.
- *Address Attestations*, welche die Zuordnung von Adressraum zu AS regeln
- *Route Attestations*, welche die Berechtigung zur Weitergabe von erhaltenen Routen regeln
- *IPsec* zur Ende-zu-Ende Verschlüsselung zwischen zwei BGP-Routern

Damit soll sichergestellt werden, dass BGP-Router untereinander sicher kommunizieren können. Ebenso wird sichergestellt, dass die übermittelten Nachrichten gültig sind. Damit kann effektiv verhindert werden, dass ein falsch konfiguriertes AS das Internet kompromittiert.

4.2 Public Key Infrastructure

Die PKI von S-BGP setzt X.509v3 mit Public Keys ein. Damit kann die Identität und Autorität von BGP-Routern, AS-Eigentümern und Adressraum-Eigentümern sichergestellt werden. [SeLK01]

4.2.1 Addressvergabe

Der Zertifikatsbaum bildet hier die derzeitige Struktur der Adressvergabe ab. Auf Root-Ebene steht die ICANN[4]. Auf zweiter Ebene folgen die regionalen Adressverwalter (z.B. ARIN[5], APNIC[6] und RIPE[7]). Danach folgen die ISPs und Kunden, welche direkt von einer Registrierungsbehörde ihren Adressraum erhalten. Bekommt ein Kunde seinen Adressraum von einem ISP zugewiesen, wird dieser Kunde auch im Baum unterhalb des ISP eingeordnet. [SeLK01]

Die Zertifikate können dabei auch mehrere getrennte Adressräume enthalten, sofern diese vom gleichen Parent bezogen werden.

4.2.2 AS-Nummernvergabe

Auch hier bildet der Zertifikatsbaum die derzeitige Struktur der Adressvergabe ab. Die erste und zweite Ebene ist identisch mit dem Baum für die Adressvergabe. Auf dritter Ebene stehen die Unternehmen, die von den regionalen Registrierungsbehörden eine oder mehrere AS-Nummern erhalten haben. Auf vierter Ebene stehen die Router, welche einer definierten AS-Nummer zugeordnet werden. [SeLK01]

[4] Internet Corporation for Assigned Names and Numbers
[5] American Registry for Internet Numbers
[6] Asia-Pacific Network Information Centre
[7] Réseaux IP Européens

4.3 Attestations

Eine *Attestation* ist eine digital signierte Beglaubigung. Man unterscheidet hier zwischen: [Kent03]

- *Address Attestations* werden vom Adressrauminhaber signiert und bestätigen, dass ein AS diesen Adressraum als eigenen Adressraum bekannt geben darf.
- *Route Attestation* werden von einem S-BGP-Router signiert und bestätigen, dass ein Nachbar-AS eine Route zum eigene AS unterhält.

4.4 Nachrichtenverifizierung

Erhält nun S-BGP-Router eine neue Nachricht, muss nun geprüft werden, ob das erste AS in der Route berechtigt ist, den Adressraum bekannt zu geben und alle nachfolgenden AS in der Route berechtigt sind, die Route weiterzugeben.

Dazu benötigt der Router die *Address Attestions*, Public Keys der AS, sämtliche *Route Attestions* und alle Public Keys der beteiligten Router auf der Route. Nachdem sämtliche Querverweise geprüft wurden und alles für gültig befunden wurde, wird die Nachricht als gültig angesehen und die normale BGP-Verarbeitung kann beginnen. [Kent03]

4.5 IPsec

Für die gesicherte BGP Kommunikation zwischen zwei S-BGP-Routern wird IPsec, genauer *Encapsulating Security Payload (ESP)* eingesetzt. Dadurch wird sowohl Authentifizierung als auch Datenintegrität sichergestellt.

4.6 Zusammenfassung

S-BGP behebt viele der Schwachstellen von BGP. Allerdings hat diese Sicherheit auch ihren Preis. Die Größe der übermittelten Nachrichten würde um ungefähr 800% wachsen. Und damit ein S-BGP-Router die übermittelten Nachrichten verifizieren kann, müssen ihm eine Menge an Daten zur Verfügung stehen. Die belaufen sich auf ungefähr 80MB für die *Address Attestations* und Zertifikate plus 30MB pro Peer für die *Route Attestions* welche im Router gespeichert werden müssen. Außerdem müssen leistungsfähige Router zur Verfügung stehen, die diese Daten zeitnah verarbeiten können.

5 soBGP

Wie auch S-BGP versucht auch das Secure Origin Border Gateway Protocol (soBGP) die Sicherheitsaspekte von BGP zu verbessern. Dabei geht es um folgende Punkte: [Whit03]

- Berechtigung des Ursprungs-AS, die Route zu diesem Adressraum bekannt zu geben.
- Prüfung, ob der Pfad zwischen dem AS des BGP Nachbarn und des Ziel-AS des Adressraums wirklich existiert.
- Einhaltung der Richtlinien des Ziel-AS des Adressraums bei der Weitergabe der Routen.

5.1 Zertifikatstruktur

Um diese Fragen beantworten zu können, existieren bei soBGP mehrere Zertifikate. [Whit03]

- Das *Entity Certificate (EntityCert)* regelt die Zuordnung zwischen einer AS-Nummer zu einem Schlüsselpaar (öffentlich/privat). Dieses EntityCert wird dabei von einer höheren Instanz signiert. Wer diese höhere Instanz sein soll, ist noch nicht abschließend geklärt. Mögliche Kandidaten sind z.B. die großen Tier-1-Provider, allgemeine Zertifizierungsunternehmen (z.B. VeriSign) oder Behörden.
- Das *Authorization Certificate (AuthCert)* regelt anschließend die Zuordnung zwischen einem AS und seinen Adressblöcken. Das Zertifikat wird hierbei von der IP-Vergabestelle (dies kann sowohl ein regionaler Registry (z.B. RIPE), aber auch ein ISP sein) signiert.
- Das *Policy Certificate (PolicyCert)* bestimmt abschließend die Richtlinien zu einem Adressblock in Bezug auf Behandlung und Weitergabe der betreffenden Routen. Das Zertifikat wird vom EntityCert des bekannt gebenden AS signiert.

Da für jedes AS normalerweise nur drei Zertifikate existieren, kann die Anzahl an Zertifikaten, die in Routern gespeichert werden müssen, gering gehalten werden. Der Austausch von Zertifikaten kann dabei direkt zwischen den BGP-Routern stattfinden, welche nach Erhalt die Signatur prüfen.

5.2 Nachrichtenverifizierung

Die Nachrichtenverifizierung kann entweder wie bei S-BGP vor der Routenauswahl stattfinden oder danach. Die Route wird dabei zunächst als gültig erachtet und ganz normal bei der Auswahl einbezogen. Dadurch sollen mögliche Verzögerungen durch den Überprüfungsvorgang vermieden werden. Anschließend wird die neue Route mittels Zertifikate überprüft und schlimmstenfalls wieder komplett entfernt. [Whit03]

5.3 Zusammenfassung

Grundsätzlich versucht soBGP die gleichen Probleme zu lösen wie S-BGP, allerdings mit einigen Unterschieden. S-BGP setzt auf eine zentrale Zertifikatsstruktur und -verwaltung. Bei soBGP hingegen kann jedes AS seine Zertifikate selbst verwalten und stellt diese ggf. im BGP-Dialog zur Verfügung. Zudem versucht soBGP, die Last auf den Routern sowie die Beeinträchtigung auf den bisherigen BGP-Prozess so gering wie möglich zu halten.

6 rpsec

Da eine allgemeine Grundlage im Bezug auf die Notwendigkeit von Sicherheit bei Routingprotokollen fehlt, wurde eine Arbeitsgruppe zu diesem Thema bei der IETF[8] gegründet. Es sollen allgemeine Vorgaben erarbeite werden, welche eine Hilfestellung für neue Router-zu-Router Protokolle darstellen.

[8]The Internet Engineering Task Force

6.1 Schutzbedürfnis

Generell gibt es zwei Bereiche, die bei BGP geschützt werden müssen: Einerseits die Verbindung zwischen zwei BGP Routern, z.B. mittels IPsec. Und andererseits muss der Inhalt der Nachrichten verifiziert werden. Hierbei werden genau die Punkte gefordert, die sowohl S-BGP als auch soBGP bereits umsetzen: [ChTa08]

- Darf das Ursprungs-AS die Route zu diesem Adressraum bekannt geben?
- Existiert der angegeben AS-Pfad wirklich im Netzwerk und wurde dieser AS-Pfad auch wirklich so von allen Beteiligten beworben?

6.2 Generelle Anforderungen

Bei sämtlichen Sicherheitsforderungen dürfen die Rahmenbedingungen eines Protokolls nicht aus den Augen verloren werden.

Ein Hauptaugenmerk liegt dabei auf der Zeit die eine empfangene BGP-Nachrichten braucht, um verarbeitet und weitergeleitet zu werden. Steigt diese zu stark an, kann das die gesamte Netzfunktionalität in Mitleidenschaft ziehen, da es zu lange dauert, bis ausgefallene Verbindungen in allen Ecken des Internets angekommen sind und bis dahin u.U. Pakete verloren gehen können. rpsec empfiehlt hier die empfangenen Nachrichten direkt nach Erhalt zu prüfen oder den gesamten Bestand an bestehenden Routen regelmäßig zu verifizieren. Hierbei sollte es möglich sein, über eine Richtlinie festzulegen, was wichtiger ist, die schnelle Routenweitergabe oder die Sicherheitsüberprüfung, welches dann den Ablauf der Routenverifizierung beeinflusst.

Zusätzlich muss ein neues Protokoll in der Lage sein, das bestehende BGP nach und nach abzulösen. Es ist schlicht nicht machbar, alle BGP-Router auf einmal zu aktualisieren und selbst innerhalb eines großen AS kann es vorkommen, dass eine zeitgleiche Umstellung nicht möglich ist. Ein neues Protokoll muss daher abwärtskompatibel mit BGP sein. Es müssen sich ebenfalls einzelne Sicherheitsfunktionen pro BGP-Nachbarn an- und abschalten lassen.

Auch darf das neue Protokoll nicht von externen Systemen abhängig sein. Existiert z.B. eine externe Verwaltung für die Sicherheitsinformationen, so muss ein gerade erst eingeschalteter Router auch ohne sofortigen Zugriff auf dieses System einen Betrieb aufnehmen können. Die Sicherheitsinformationen werden hierbei dann durch eine spätere Synchronisation aktualisiert. Dadurch soll die Anlaufzeit nach einem Ausfall möglichst gering gehalten werden.

Da durch die viele Verschlüsselung der Arbeitsaufwand der Router sehr stark ansteigen kann, sollten umfangreiche Tests mit aktuell eingesetzter Routerhardware durchgeführt werden, um die Auswirkungen schon im Vorfeld abschätzen zu können.

Um die Administration einfach zu gestalten, wäre es wünschenswert, wenn die Konfiguration auf den Routern so kurz und so einfach wie möglich gehalten wird. Zudem sollten, wenn möglich, Einsteiger mit dem Protokoll direkt loslegen können, ohne vorher Hunderte Seiten Dokumentation lesen zu müssen. [ChTa08]

6.3 Sicherheitsinfrastruktur

Falls das Protokoll eine Sicherheitsinfrastruktur zur Verwaltung der Authentifizierung- und Verifizierungsdaten einsetzt, so muss diese gewisse Anforderungen erfüllen. Dazu zählt die Robustheit gegen äußere Angriffe, sowie die dauerhafte Integrität der verwalteten Daten. Nach Möglichkeit sollte diese Struktur keine neuen Verwaltungsorganisationen einführen, sondern z.B. auf 1. und 2. Ebene sich auf vorhandene Strukturen beziehen.

6.4 Routenüberprüfung

Erhält ein BGP-Router eine neue Nachricht, soll dabei folgendes geprüft werden: [ChTa08]

- Ist das Ursprungs-AS vom Adressrauminhaber berechtigt, den Adressraum bekannt zu geben?
- Ist das nächste AS im AS-Pfad auch wirklich das AS, von dem die BGP-Nachricht erhalten wurde?
- Existieren sämtliche ASe des AS-Pfades und haben diese auch wirklich eine Verbindung zueinander?
- Nahm die Routenbekanntgebung auch wirklich den Pfad, den der AS-Pfad abbildet? Dazu kann es erforderlich sein, dass jeder beteiligte Router seine Existenz durch einen Anhang an der BGP-Nachricht bestätigt. Wie dies genau aussehen soll, ist allerdings noch nicht abschließend geklärt.

Bei all diesen Maßnahmen soll es für den Administrator möglich sein, zu entscheiden, bei welchen BGP-Nachbarn welche Überprüfungen durchgeführt werden. Zudem muss sichergestellt werden, dass die dazu notwendigen Daten den Routern zeitnah zur Verfügung stehen, um die BGP-Nachrichten ohne Wartezeit zu verarbeiten.

6.5 Zusammenfassung

rpsec beschreibt sehr umfangreich die Anforderungen an den Nachfolger von BGP. Es bleibt zu hoffen, dass es nicht mehr lange dauert, bis ein Nachfolger gefunden wird, welcher BGP ablöst. Die Rahmenbedingungen dafür sind jedenfalls gesteckt.

7 Der Fall YouTube

Ende Februar wurden die Provider von Pakistan von der Regierung angewiesen, YouTube zu sperren, da dort ein anti-koreanisches Video aus den Niederlanden abrufbar war. Zum AS36561 (YouTube) gehört u.a. der Adressraum 208.65.152.0/00. Das Routing für diesen Adressraum ist in Abb. 5 dargestellt und wurde von der gesamten Aktion nicht beeinflusst.

Am 24. Februar 2008 um 18:47 Uhr begann nun das AS17557 (Pakistan Telecom) den Adressraum 208.65.153.0/24 bekannt zu geben, da sich in diesem Netz die IPs der Webserver befinden. Da dieses 24er Netz spezieller als das 22er Netz von YouTube ist, wird es bei BGP gegenüber dem 22er Netz bevorzugt. Da der Provider von Pakistan Telecom, AS3491 (PCCW Global), die BGP-Nachrichten ungefiltert weiterleitet, war die Route innerhalb von 2min fast im ganzen Internet bekannt und wurde als beste Route zum Ziel ausgewählt wie Abb. 6 zeigt.

Um 20:07 Uhr begann jetzt YouTube ebenfalls, das 24er Netz bekannt zu geben, was darin resultierte, dass ein Teil des Traffics nach Pakistan und ein Teil zu YouTube ging, da es den Routern überlassen war, welche Route sie bei gleicher Netzgröße wählen. Um 21:01 Uhr begann PCCW Global damit, die Routen über das YouTube-Netz von Pakistan Telecom zu widerrufen. Um 21:23 Uhr kehrte dann wieder Ruhe ein und praktisch der gesamte Traffic ging wieder an YouTube wie Abb. 7 zeigt. [RIS08]

8 Zusammenfassung

Obwohl immer wieder Probleme mit BGP auftreten, hat sich das Protokoll als sehr robust herausgestellt. So federt es Ausfälle von Internetstrecken sehr gut ab. Doch beim Design

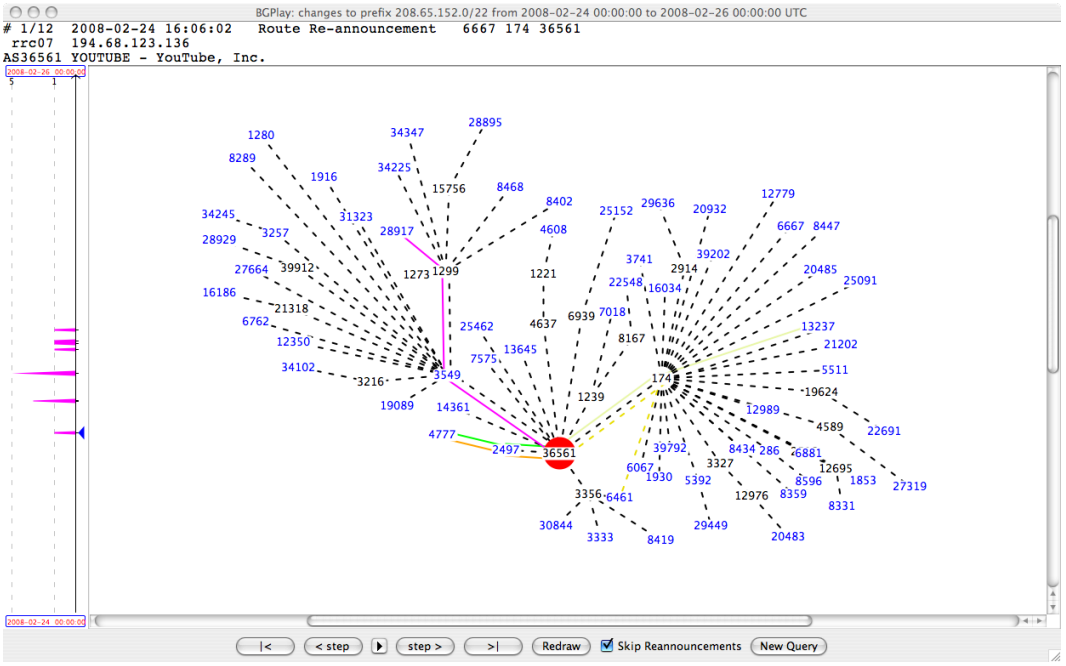

Abbildung 5: Routing für 208.65.152.0/22

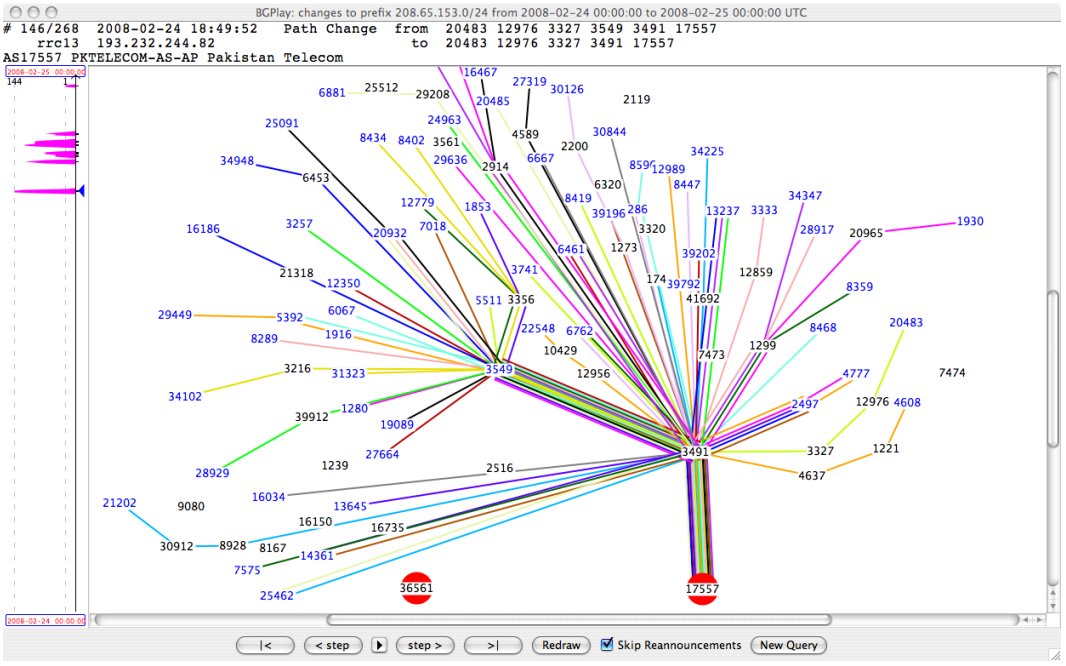

Abbildung 6: Routing für 208.65.153.0/24 um 18:49 Uhr

des Protokolls stand die Funktion und nicht die Sicherheit im Vordergrund. Damals war die Struktur des Internet noch deutlich kleiner und man kannte alle Beteiligten. Heute hingegen ist das Internet riesig und es existieren Tausende von ASen. So kommt es immer wieder zu gewollten oder ungewollten Beeinträchtigungen. Allerdings muss ein Angreifer erstmal an den Kern des Internets herankommen, was zumindest eine Abschottung gegen „Script-Kiddies“ bietet. Es sollte sich bald ein Protokoll, möglichst auf Grundlage von rpsec, finden, welches

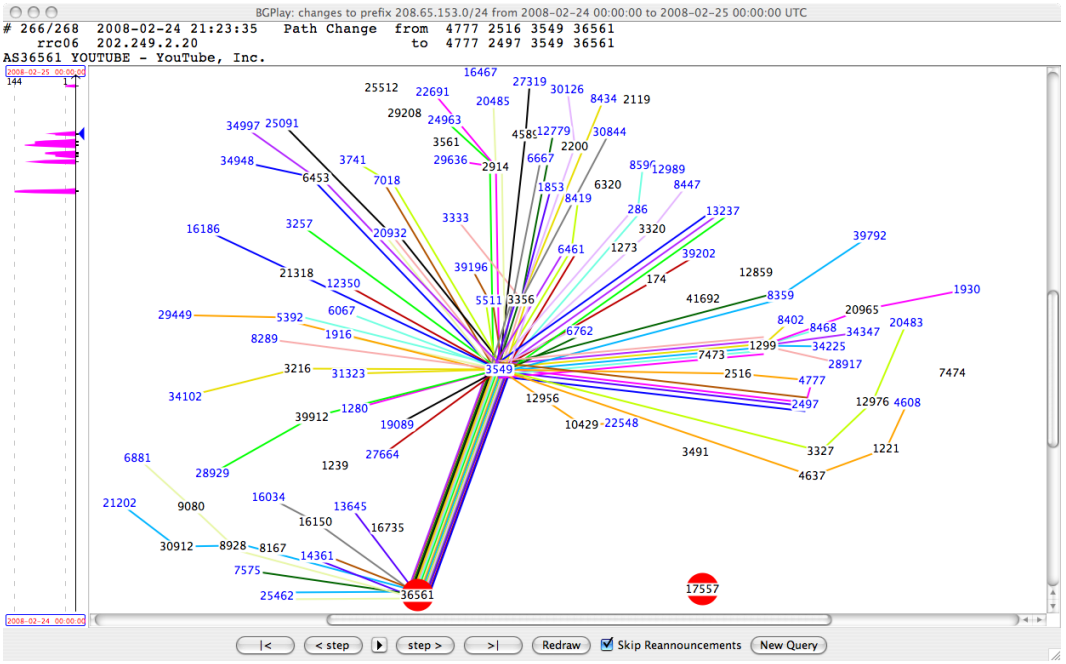

Abbildung 7: Routing für 208.65.153.0/24 um 21:23 Uhr

den Kern des Internets wieder ein Stück sicherer macht, denn das Wachstum ist noch lange nicht abgeschlossen.

Literatur

[Beij02] Iljitsch van Beijnum. *BGP*. O'Reilly. 1. ed.. Auflage, 2002.

[ChTa08] B. Christian und T. Tauber. BGP Security Requirements (draft-ietf-rpsec-bgpsecrec-10), 2008.

[KeLS00] Stephen Kent, Charles Lynn und Karen Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications* 18(4), 2000, S. 582–592.

[Kent03] Stephen Kent. Securing the Border Gateway Protocol. *The Internet Protocol Journal* 6(3), 2003.

[PiKa08] Alex Pilosov und Tony Kapela. Stealing The Internet. Technischer Bericht, Defcon 16, 2008.

[RIS08] Routing Information Service RIS. YouTube Hijacking: A RIPE NCC RIS Case Study. RIPE NCC News, 2008.

[SeLK01] Karen Seo, Charles Lynn und Stephan Kent. Public-Key Infrastructure for the Secure Border Gateway Protocol (S BGP). *DARPA Information Survivability Conference and Exposition* Band 1, 2001.

[Whit03] Russ White. Securing BGP Through Secure Origin BGP. *The Internet Protocol Journal* 6(3), 2003.

Abbildungsverzeichnis

Useful Network Security

Daniel Lienert

Sichere Übertragung, Authentifizierung und Signierung sollten heutzutage unter den Internetbenutzern ein wohlverstandenes und selbstverständliches Thema sein, findet dieses Medium doch mehr und mehr Akzeptanz für sensible Transaktionen wie Online Shopping und Online Banking. Doch verschiedene Studien zeigen, wie schwer das Thema Sicherheit dem Anwender fällt und machen dafür die komplizierten und irreführenden Oberflächen der Sicherheitsanwendungen verantwortlich. In der folgenden Arbeit soll nun in einzelnen Beispielen auf die Probleme der Anwendungen eingegangen, die Ursachen analysiert und einige theoretische und praktische Verbesserungsmöglichkeiten aufgezeigt werden.

1 Einleitung

Seit der Kommerzialisierung und der öffentlichen Verfügbarkeit in den Jahren 1991 – 1993, entwickelte sich das Internet aus einem Medium für Naturwissenschaftlern und Informatikern zu einer Allgegenwärtigen Informations- und Kommunikationsplattform, welche heute eine breite Bevölkerungsschicht erreicht.

Stand Juni 2008 verwenden ≈50% der Europäer, was ca. 385 Millionen Nutzern entspricht, beruflich oder privat das Internet [Grou08]. Laut der Studie „Entwicklung der Informationsgesellschaft – IKT in Deutschland – Ausgabe 2007“ [Deut07] des statistischen Bundesamtes sind 61,4% der Haushalte in Deutschland an das Internet angeschlossen, 50% davon bereits mit DSL oder äquivalenten Breitbandverbindung.

Die beliebtesten Anwendungen des Internets, so zeigt die Studie weiterhin, sind mit einem Anteil von 85% der Internetnutzer das Versenden und Empfangen von eMails, gefolgt von der Suche nach Informationen über Waren und Dienstleistungen mit 83%. Bereits 45% der Männer und 39% der Frauen verwenden das Internet für Online Banking.

Während sich das Internet somit einerseits aus der Domäne der Informatiker und technisch versierten Benutzer, hin zum Plug&Play-Netz für Jedermann entwickelt, gewinnt es auf der anderen Seite mehr und mehr an Bedeutung als Marktplattform für sicherheitskritische Transaktionen in Online Shops und Online Banking.

Während also die Designs der Shopoberflächen immer bunter und eleganter, die Interfaces der Online- Banking Angeboten immer einfacher benutzbar werden, entziehen sich die zugrunde liegenden Techniken immer mehr dem Verständnis des gemeinen Internetnutzers. Fatal wird diese Unkenntnis dadurch, dass sich nur wenige Surfer mit den Sicherheitsaspekten und dem Schutz ihrer Daten befassen. Die Auswirkungen davon gehen als Phishing-Angriffe, Identitätsdiebstahl und Passwortklau immer wieder durch die Presse.

Eine Ursache dieses Problems ist im benutzerunfreundlichen Design der Sicherheitsfunktionen der Browser und eMail-Clients zu suchen, welche offensichtlich nicht für den Gelegenheitssurfer ausgelegt sind und Diesen durch undurchsichtige Konfigurationen und nicht zielführende Informationen von der Benutzung abschrecken.

Im folgenden Kapitel wird auf das Design der Sicherheitsfunktionen der beliebtesten Anwendungen, WWW und eMail eingegangen, Mankos aufgezeigt und Verbesserungsmöglichkeiten betrachtet.

2 Was ist Sicherheit?

Die Sicherheit einer Applikation ist immer ein Kompromiss zwischen der theoretisch möglichen und der in der Praxis akzeptierten und einsetzbaren Sicherheit. In der Theorie ist der Einsatz einer PKI[1] und einer Authentifizierung via Chipkarte, der einfachen Authentifizierung mittels Benutzername und Passwort vorzuziehen. In der Praxis aber ist diese sichere Methode meist die entscheidend aufwendigere und im Unternehmensumfeld oft mit beträchtlichen Kosten verbunden. Doch die sichere Variante ist nutzlos, sobald die Anwender den Aufwand scheuen und es vorziehen keine Sicherheitsfunktionen zu verwenden.

Hinzu kommt, dass für Kryptologen und Sicherheitsexperten meist nur die perfekte Sicherheit akzeptable ist. Mit diesem Alles-oder-Nichts-Ansatz riskieren sie aber, dass der Anwender sich im Zweifelsfall für „nichts“, also für Verzicht auf jegliche Sicherheit, entscheidet.

3 Sicherheit in Internetanwendungen

Nur wenige Programme zur Internetkommunikation werden geschrieben, deren primäre Aufgabe Sicherheitsfunktionen darstellen. Viel häufiger sind Programme, für die Kryptographie eine Erweiterung ihrer Hauptaufgabe beziehungsweise eine Erweiterung der verwendeten Protokolle darstellt. Genauer sollen im Folgenden die zwei beliebtesten Internetanwendungen WWW mit der HTTP[2] Erweiterung HTTPS[3]/TLS[4] und eMail mit dem Einsatz von PGP[5] zur Signierung und verschlüsselten Übertragung von eMails betrachtet werden.

Zwar ist die Wichtigkeit der sicheren Anwendungsprogrammierung mittlerweile in den Köpfen der Programmierer angekommen, trotzdem wurden und werden die Benutzerinterfaces der Sicherheitsfunktionen eher stiefmütterlich behandelt.

3.1 Sicherheit im Webbrowser

Alle Bestrebungen das Surfen im Netz, Online Shopping und Online Banking sicher zu machen sind vergebens, wenn der Anwender nicht für die Gefahren im Web sensibilisiert werden kann, so dass er Gefahren richtig einschätzt und entsprechend reagiert.

Doch „eine praktisch unwirksame Maßnahme ist das Schulen der Benutzer“ schreibt Peter Gutmann 2006 in seinem Artikel „Nur für Spezialisten – Sicherheitssoftware in der Kritik“ für das Linux-Magazin [Gutm06]. Die Anwender sind bereits falsch „konditioniert“. Sie sind gewohnt, während ihrer Arbeit im Web ständig mit Fehlermeldungen, Warnungen und Popup-Meldungen zu DNS oder Webserverfehlern, fehlenden Plugins oder abgelaufenen Zertifikaten konfrontiert zu werden.

[1] Public-Key-Infrastruktur, System zur Aussstellung, Verteilung und Verifikation von Zertifikaten zur Absicherung computergestützter Kommunikation.

[2] Hypertext Transfer Protocol, Protokoll der Anwendungsschicht zur Übertragung von Daten im World Wide Web.

[3] Hypertext Transfer Protocol Secure – HTTP over TLS.

[4] Transport Layer Security – symetrisches Ende-zu-Ende-Verschlüsselungsverfahren.

[5] Pretty Good Privacy. Programm zur Signierung und Verschlüsselung von Daten.

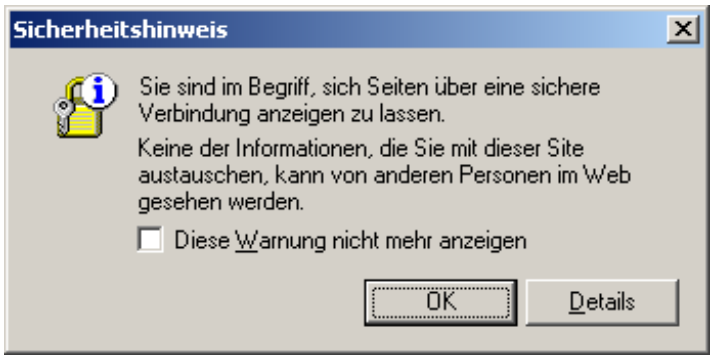

Abbildung 1: Internet Explorer 6 warnt den Benutzer vor einer sicheren Verbindung.

Ein Beispiel für solch eine Meldung liefert uns der Internet Explorer 6 in Abbildung 1. Die Meldung erscheint vor der Verbindung mit einer SSL geschützten Seite und warnt den Benutzer davor, daß seine Daten nun nicht mehr von Fremden gelesen werden können.

So kommt es, dass der Benutzer nicht zwischen kritischen und unkritischen Fehlermeldungen unterscheidet. Warnungen vor falschen oder abgelaufenen SSL-Zertifikaten beim Online Banking werden wie die vorherigen, unkritischen Fehlermeldungen behandelt und bedenkenlos bestätigt. Die Psychologen nennen dieses Verhalten Urteilheuristik: Statt immer neu über ein Problem nachzudenken, wendet der Benutzer eine Entscheidung an, welche schon früher zu einem guten Ergebnis führte, auch wenn er mit diesem Verhalten ab und an falsch liegt. Dieses Phänomen mag erklären, warum von 300 Kunden einer Bank, welche ein falsches Zertifikat auf ihrem Server eingesetzt hatte, nur ein Kunde die Verbindung ablehnte während die Übrigen die Warnung des Browsers ignorierte.

3.2 Richtlinien für ein benutzerfreundliches Interfacedesign

Solches Benutzerverhalten rührt zu einem großen Teil von einem schlechten und lieblosen Anwendungsdesign her. Dabei reichen einige wenige Änderungen und die Ausrichtung am Benutzerverhalten um diesen Umstand entscheidend zu verbessern. Peter Gutmann zeigt in seinem Bericht für das Linux-Magazin einige Vorschläge, welche hier nun kurz zusammengefasst werden:

Das Verwenden eines Programms dient meist dem Erledigen einer Aufgabe. Funktionen wie Sicherheitsfeatures, welche nicht unbedingt auf dem Weg zum gewünschten Ziel liegen, werden ignoriert – Warnmeldungen und Abfragen dazu weg geklickt. Ein Ansatz ist, die Sicherheitsfunktionen derart zu integrieren, dass sie quasi im Huckepack-Verfahren auf dem Lösungsweg angewendet werden müssen. Als Beispiel nennt der Autor den Zündschlüssel eines Autos, der zwar ein integraler Bestandteil des Sicherheitssystem ist, für den Anwender aber nur Mittel zum Zweck: das Auto zu Starten.

Eine weitaus einfachere Maßnahme mit großer Wirkung ist die Anpassung der Dialoge. Obwohl in der Vergangenheit von Anwendungsentwicklern verpönt, sollten alle Dialoge für sicherheitsrelevante Nachfragen modal ausgelegt sein.

Des Weiteren sollten die Dialoge so gestaltet sein, dass sie die Aufmerksamkeit des Benutzers auf sich ziehen und ihn so dazu bringen seine Entscheidung aktiv zu treffen. Eine Maßnahme dazu ist das Deaktivieren des Schließenknopfes in der rechten oberen Dialogecke. Außerdem sollten der beschreibende Text sowie die Beschriftung der Buttons aussagekräftig sein. Der Warnhinweis zu einem ungültigen SSL-Zertifikat des Internet Explorers 6 von Microsoft in

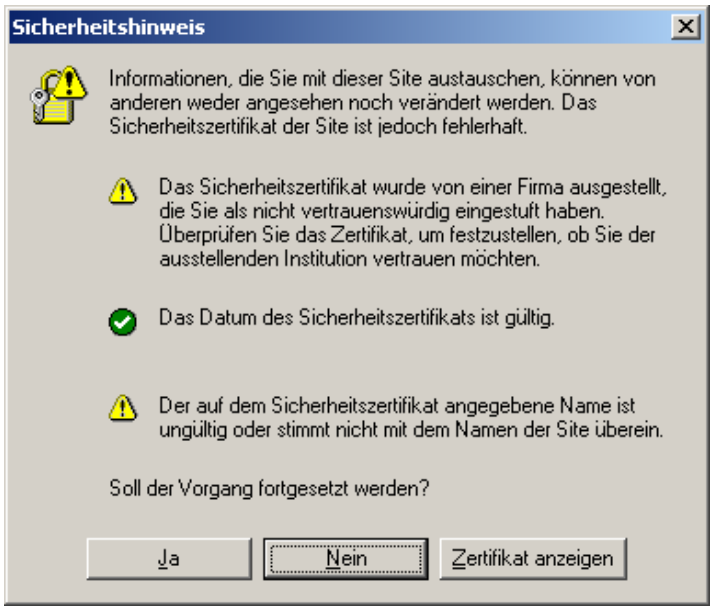

Abbildung 2: Der Dialog liefert keine Informationen über die möglichen Risiken. Der Schließen Knopf und die Standard-Buttons laden zum wegklicken ein.

Abbildung 2 zeigt uns ein gutes Negativ-Beispiel. Ein Anwender würde ohne auf die Warnung einzugehen, den Schließen- oder Ja-Knopf betätigen, um schnellstmöglich seine gestellte Aufgabe zu erledigen.

Dr. Ka-Ping Yee stellt in seiner oft zitierten Arbeit „User Interaction Design for Secure Systems“ [Yee02] zehn Richtlinien zu einem anwenderfreundlichen Design für Sicherheitsapplikation auf. Die im folgenden aufgezählten Richtlinien sind eine Übersetzung der aktualiserten Fassung auf Yee's Internetseite [Yee05].

1. *Weg des geringsten Widerstands.* Der komfortabelste und einfachste Weg etwas zu tun, sollte auch der sicherste sein.
2. *Aktive Autorisation.* Rechte an andere (Applikationen) werden nur vergeben, nachdem dies dem Benutzer signalisiert wurde und dieser das aktiv bestätigt.
3. *Widerufsmöglichkeit.* Gib dem Benutzer die Möglichkeit, die Rechte Anderer auf seine eigenen Ressourcen zu beschränken.
4. *Sichtbarkeit.* Biete dem Benutzer eine genaue Einsicht, welche Auswirkungen seine Entscheidung auf die Rechte Anderer hat.
5. *Bewusstsein.* Mach dem Benutzer die eigenen Zugriffsrechte auf Ressourcen bewusst.
6. *Sicherer Kanal.* Schütze die Übertragungskanäle des Benutzers, so dass es einem Angreifer nicht möglich ist, Berechtigungen im Namen des Benutzers zu ändern.
7. *Aussagekraft.* Gib dem Benutzer die Möglichkeit, eine Sicherheitsrichtlinie zu den Bedingungen zu definieren, welche für seine Aufgaben nützlich ist.
8. *Relevante Grenzen.* Visualisiere Unterscheidungen zwischen Objekten und Aktionen entlang den Grenzen, welche für die Aufgabe des Benutzers relevant sind.
9. *Identifizierbarkeit.* Verwende für Objekte und Aktionen klar erkennbare und eindeutige visuelle Repräsentationen.
10. *Vorschau.* Zeige die Konsequenzen einer Benutzerentscheidung klar und verständlich auf.

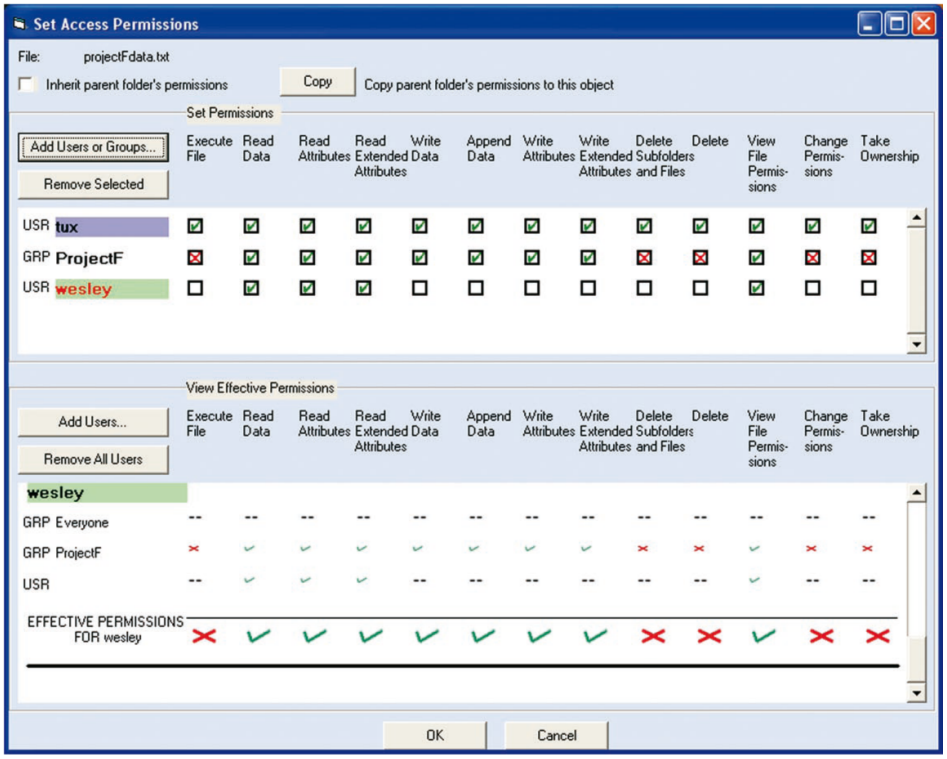

Abbildung 3: Basierend auf den von Salmon gezeigten Informationen kann der Benutzer die richtigen Sicherheitsentscheidungen treffen. Quelle: [Payn08]

Natürlich garantiert die Einhaltung dieser Regeln nicht automatisch ein besseren Anwendungsdesign. „Würden solche Regeln existieren, würden wir diese ganz sicher alle verwenden und das Usability-Problem wäre gelöst.“

Bryan D. Payne und W. Keith Edward vom Georgia Institute of Technology evaluieren in ihrem Artikel „A Brief Introduction to usable Security“ [Payn08] anhand dieser Regeln innovative Interfaces und setzen diese mit der Akzeptanz der Benutzer in Beziehung.

Eines der dort untersuchten Interfaces ist das Salmon Interface[Robe05] für die Erteilung von Dateirechten unter Windows. Während die Zuteilung von Rechten unter Windows durch die vielen Möglichkeiten von Benutzer und Gruppenregeln recht komplex ist und das hauseigene Interface kaum Möglichkeiten gibt sich die effektiven Dateirechte pro Datei anzeigen zu lassen, bietet die Salmon Oberfläche alle relevanten Informationen in einer aufgeräumten Oberfläche an. Robert Reeder und Roy Maxion, Entwickler der Oberfläche stellen in ihren Test eine 4-fach höhere Genauigkeit und eine Reduzierung der Fehleinstellungen 94% in ihrer Benutzerstudie fest.

Testet man das Interface gegen Yee's Richtlinien, zeigt sich, dass Salmon alle Regel bis auf eine betrachtet. Es zeigt alle relevanten Operationen in einem Interface und bietet somit Sichtbarkeit, Bewusstsein, Aussagekraft und Vorschau. Da nur die Rechte eines Objektes bearbeitet werden, bietet es auch Identifizierbarkeit und relevante Grenzen. Durch die erstaunliche Genauigkeit, mit welcher die Benutzer arbeiten konnten, bietet das Interface auch den Weg des geringsten Widerstandes.

3.3 Verbesserungen in der Praxis

Was Peter Gutmann im Jahr 2006 in seinem Artikel hinsichtlich der Schwächen in den Dialogen der Webbrowser bemängelt blieb indes nicht ungehört. Sowohl der Platzhirsch Microsoft Internet Explorer als auch Open-Source-Browser wie Mozilla Firefox haben ihre Lektion gelernt und eifrig nachgebessert.

Abbildung 4: Ein kräftiger, grüner Balken zeigt nun ein verifiziertes SSL-Zertifikat an.

Farben spielen eine große Rolle beim Erkennen und Bewerten von Gefahrensituationen und das Ändern der Hintergrundfarbe oder des Rahmens eines Objektes eignet sich als effektives Kommunikationsmittel. Während Mozilla Firefox schon in der Version 2/2006 die Validität des SSL Zertifikat einer HTTPS-Verbindung farblich hervorhob, wird in der aktuellen Version in einem auffälligen grünen Feld zusätzlich der Name des Zertifikatbesitzers eingeblendet, falls es sich um ein „Extended Validation SSL-Zertifikat“ handelt. Bei dieser Art von Zertifikaten erfolgt eine genaue Identitätsüberprüfung des Domaininhabers durch den Zertifikatsaussteller.

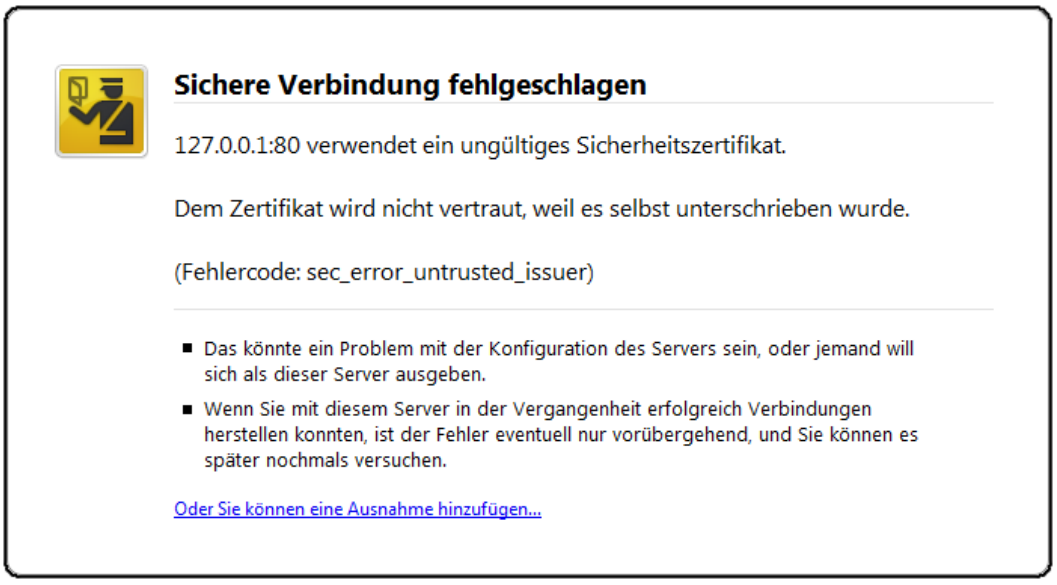

Abbildung 5: Die Dialoge der aktuellen Browser sind bereits weitaus aussagekräftiger.

Eine weitreichende Verbesserung zeigt sich bei der Anzeige des Fehlerfalls. Zunächst wird die Fehlermeldung nicht mehr als Pop-Up-Dialog, sondern im Hauptbereich des Browsers angezeigt. Die Fehlermeldung zeigt in den ersten zwei Zeilen nach der Überschrift was für ein Fehler auftrat und was der Grund für die Meldung ist. Darunter wird in zwei Stichpunkten prägnant und verständlich die Gefahren für und die Möglichkeiten des Benutzers erklärt.

Der eigentliche Clou: Statt dem Benutzer eine Variation der beliebigen Schaltflächen „Ja“, „Nein“ oder „Abbrechen“ zu bieten, wird hier in einem eher unauffälligen Link dem Benutzer erklärt, dass er eine Ausnahme hinzufügen kann. Der damit verbundene Aufwand macht dem Anwender die Brisanz der Meldung klar. Der Weg des geringsten Widerstands wäre hier sicher den Vorgang abzubrechen.

3.4 eMails verschlüsseln und signieren mit PGP

Mit einem Anteil on 85% ist eMail die wichtigste Anwendung im Internet. Doch bietet das zum Mailtransfer verwendete Protokoll SMTP – Simple Mail Transfer Protokol per se keine Möglichkeit zu verschlüsselten Übertragung noch, ist eine Signierung des Inhalts der Nachrichten vorgesehen. Verschlüsselung und Signierung per PGP ist eine der populärsten Möglichkeiten diese Lücke zu schließen. Trotzdem ist die Anwendung von PGP kaum verbreitet.

Alam Whitten und J.D. Tyger untersuchen in ihrer Arbeit „Why Johny Can't encrypt: A Usability Evaluation of PGP 5.0" [WhTy99] die Ursachen der geringen Akzeptanz. Whitten und Tyger erkennen grundsätzlich 5 Schwierigkeiten, denen sich das Interfacedesign stellen muss um eine nutzbare Oberfläche für Sicherheitsanwendungen abzugeben:

Die Schwierigkeit des unmotivierten Benutzers: Sicherheit ist selten ein primäres Ziel der Anwender sondern eine Nebenaspekt, welches die Anwender schützt, während sie ihren Hauptaufgaben, wie surfen und eMail versenden nachgehen. Daher müssen sich die Anwendungsentwickler der Tatsache bewusst sein, das Benutzer immer annehmen, geschützt zu sein und in keinem Fall nach versteckten Sicherheitsoptionen suchen.

Das Problem der Abstraktion: Computersicherheit basiert oft auf komplexen, abstrakten Regelsätzen um den Zugriff auf bestimmte Ressourcen zu regeln. Was für den Programmierer eine alltägliche Sache, ist für den Anwender oft befremdlich und unverständlich.

Das Problem der fehlenden Rückmeldung: Um gefährliche Fehler zu vermeiden, ist eine gute Rückmeldung zum Anwender notwendig. Diese Rückmeldung ist aber oft sehr komplex und daher schwer zu realisieren.

Das Scheunentorproblem: Ein Sprichwort besagt, dass es sinnlos ist das Scheunentor zu schließen nach dem das Pferd entfloh. Übertragen: Sobald ein Geheimnis einmal auch nur für einen kurzen Moment ungeschützt ist, gibt es keinen Weg sicher zu gehen, dass es nicht bereits gestohlen wurde. Darum muss das Design einer Sicherheitsanwendung sicherstellen, dass der Benutzer versteht was er tut, um diesen möglichen Fehlern zu bewahren.

Das Problem des schwächsten Gliedes: Da es bekannt ist, dass die Sicherheit eines vernetzten Computers nur so stark ist wie seine schwächste Komponente, muss der Benutzer dabei unterstützt werden, alle Aspekte der Sicherheit im Blick zu behalten.

Die Autoren stellten sich nun die Frage: Wenn ein durchschnittlicher Mail-Benutzer mehr Privatsphäre wünscht und zu diesem Zweck PGP einsetzten möchte, ist er mit PGP's Design in der Lage herauszufinden wir er dies bewerkstelligen kann, ohne gefährliche Fehler zu machen?

Diese Frage sollte zum einen durch eine objektive Betrachtung der Anwendung und zum anderen durch einen Laborversuch mit Testkandidaten beantwortet werden.

3.4.1 Objektive Betrachtung der Anwendung

Die objektive Betrachtung zielte zunächst auf die Anwendung PGP-Tools und dort auf die Frage, mit welchen Grafiken und Metaphern die einzelnen Aktionen: verschlüsseln, signieren, entschlüsseln und Signatur prüfen belegt wurden. Die Entwickler verwendeten hier als Metaphern Schlüsseln und ein, mit einem Schloss versehenen Brief für die Verschlüsselungsfunktion. Dabei beschreibt die Metapher aber nicht den Umstand, dass zum Ver- und Entschlüsseln unterschiedliche Schlüssel, den Public- und Private-Key, verwendet werden müssen. Gerade dies sollte dem unkundigen Benutzer aber verdeutlicht werden, weicht es doch vom alltäglichen Verständnis eines Schlüssels ab.

Weiterhin kritisierten die Autoren den Schlüsselaustausch mit den Schlüssel-Servern. Diese wichtige Standardaufgabe der Anwendung war nicht im Hauptmenü vorhanden und beim

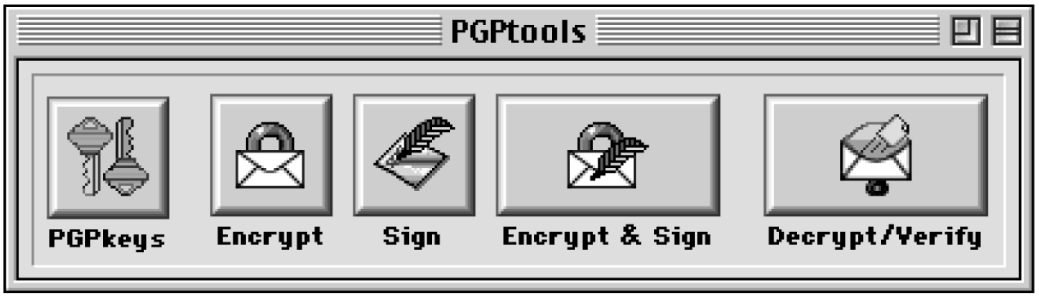

Abbildung 6: Buttonleiste der Anwendung PGP-Tools. Quelle: [WhTy99]

Schlüsselaustausch erhielt der Benutzer kein Feedback über den Status der Kommunikation, oder mit welchem Key-Server kommuniziert wurde.

Während der Arbeit mit PGP begegnet der Benutzer vielen irreversiblen Aktionen, welche von der Applikation entsprechend gekennzeichnet werden sollte. Hier erkannten die Autoren aber einige Fallen, welche auch erfahrene Benutzer betreffen können.

Versehentliches Löschen des privaten Schlüssels. Während ein versehentlich gelöschter, öffentlicher Schlüssel leicht wieder vom Keyserver geladen werden kann, ist ein gelöschter privater Schlüssel unwiederbringlich verloren. PGP reagiert auf den Löschbefehl mit der Frage „Möchten Sie diesen Eintrag wirklich löschen“ – ohne auf die Konsequenzen einzugehen.

Versehentliches Publizieren. Schlüssel können zwar einfach zu einem Keyserver hinzugefügt werden, aber dort nicht mehr gelöscht werden. Selbst wenn ein Schlüssel dort mittels eines Revoke-Zertifikats widerrufen wird, bleibt er auf dem Server sichtbar. Dieses Konzept sollte für einen unbedarften Benutzer eher unüblich sein. Trotzdem warnt die Anwendung nicht ausreichend vor den Konsequenzen eines falsch publizierten Schlüssels.

Schlüssel zurückrufen. Eigene Schlüssel können mittels eines Revocation-Zertifikats wiederufen werden. Falls das Sicherungspasswort des privaten Schlüssels verloren geht, ist dies die einzige Möglichkeit, den Schlüssel für ungültig zu erklären. Dazu muß das Zertifikat aber vorhanden sein und eine Erstellung dieses Zertifikat ist nur mit dem privaten Schlüssel und dem zugehörigen Sicherungspasswort möglich. Die Autoren gehen davon aus, dass nur wenige der Anwender das Prinzip des Widerrufszertifikats verstehen und die Anwendung gibt hier auch wenig Hilfestellung.

3.4.2 Anwendertest

Die Aufgaben des Testszenarios, welchen sich die Probanten stellen mussten, beinhalten die typischen Funktionen, mit denen ein Anwender bei der Arbeit mit PGP konfrontiert wird. Zunächst sollte ein Schlüsselpaar erzeugt werden und den öffentlichen Teil per Key-Sever oder eMail an die anderen Teilnehmer verteilt werden. Gleichzeitig sollten die Keys der übrigen Teilnehmer gesammelt werden. Danach sollte eine kurze Nachricht mit dem eigenen privaten Key signiert und mit den gesammelten Public-Keys an die übrigen Mitarbeiter verschlüsselt gesendet werden. Die Probanten wurden vor Beginn des Tests für das verwendete eMail-Programm Eudora geschult und ihnen standen die Gebrauchsanweisungen sowohl von Eudora als auch PGP 5.0 zu Verfügung; ein Testdurchlauf dauerte 90 Minuten.

Das Ergebnis der Studie war ernüchternd. Drei der 12 Probanten mailten das Geheimnis aus Versehen im Klartext, 7 verwendeten ihren eigenen öffentlichen Schlüssel um die Nachricht zu verschlüsseln, nur wenige verstanden den Zusammenhang von privatem und öffentlichem

Schlüssel. Auch das Entschlüsseln bereitete Probleme: Von den 5 Teilnehmern, welche eine korrekt verschlüsselte eMail erhalten hatten, konnten nur zwei diese auch entschlüsseln.

Auch der Schlüsselaustausch bereitete Probleme. 10 Probanten waren fähig ihre Schlüssel zu veröffentlichen, einige per Key Server, andere per direkter eMail. Große Verunsicherung herrschte bei der Schlüssel-Server Variante, einige befürchteten versehentlich ihren privaten Schlüssel zu veröffentlichen. Acht der Zwölf konnten die öffentlichen Schlüssel der anderen Teilnehmer per Key-Server importieren – fünf von ihnen aber erst nach expliziter Anweisung des Testleiters.

Das Ergebnis bekräftigte die Autoren in der Vermutung, dass das von PGP 5.0 bereitgestellte Interface nicht ausreichend ist um Computersicherheit auch für diejenigen nutzbar zu machen, welche nicht bereits mit dem Thema vertraut sind. Nur ein Drittel der Teilnehmer konnten den Test absolvieren, ein Viertel von ihnen verschickten ihren privaten Schlüssel während der 90 Minuten. Als Hauptproblem wurde gesehen, dass das System von öffentlichen und privaten Schlüsseln von den Teilnehmern nicht vollständig durchdrungen wurde und auch von der Anwendung nicht nahe gebracht werden konnte.

3.5 Die Situation knapp zehn Jahre später

Abbildung 7: Die aktuelle Oberfläche des freien GnuPG Programm Kleopatra 2008

Wie weiter oben gesehen, haben sich die Entwickler bei der Präsentation von Warnungen in Webbrowsern einige Gedanken gemacht und viele der, von Peter Gutmann 2006, erkannten Fehlern beseitigt.

Im Fall von PGP sind dagegen kaum Fortschritte zu sehen. Noch immer hat das Verschlüsseln und Signieren von eMails nicht die breite Öffentlichkeit erreicht. Das kann zum einen daran liegen, dass PGP, im Gegensatz zu HTTPS/SSL im Webbrowser, keine Bedeutung in der Arbeitsumgebung des normalen Anwenders hat.

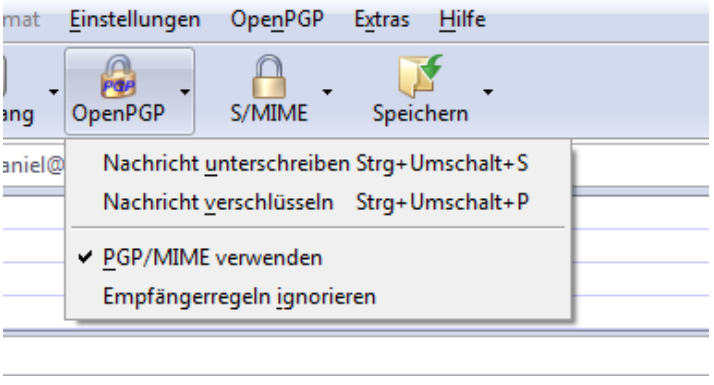

Abbildung 8: PGP Frontend als Plugin für Thunderbird.

Es gibt kaum einen Fall, in dem der Anwender gezwungen ist eine eMail zu verschlüsseln oder zu signieren um ein Ziel, etwa Einkauf in einem Onlineshop zu erreichen. Zum anderen ist diese Funktion in kaum einem populären eMail-Programm standardmäßig verfügbar. Zwar kann die Funktion in den meisten Fällen per Plug-In nachgerüstet werden, was aber einen großen Aufwand darstellt, welchem sich der Benutzer ohne fremden Antrieb kaum stellen wird. Ein kurzer Test der Oberfläche der freien PGP-Implementation HnuPG / Kleopatra zeigt dass sich hier in den letzten Jahren hinsichtlich der Usability wenig getan hat.

4 Sicherheitsprotokolle abseits der Applikationsebene

Das Einrichten von gesicherten Übertragungskanälen und die damit verbundene schwierige Aufgabe, aus der Fülle an verfügbaren Verschlüsselungsprotokollen das geeignete zu wählen, ist nicht nur eine Domäne der IT-Profis.

4.1 Lösungen für einfache WLAN Einrichtung

Schon bei der Einrichtung einer simplen WLAN(Wireless Local Area Network)-Strecke muss auch der Gelegenheitsanwender zwischen einer großen Anzahl an möglichen Verschlüsselungsoptionen, wählen. Mit der Entscheidung ob WEP[6] in 64 oder 128 Bit, offener oder gemeinsamer Authentifizierung, WPA[7] oder WPA2 mit einer AES[8]- oder TKIP-Verschlüsselung ist auch der ambitionierte Anwender schnell überfordert. Dies führte in der Vergangenheit dazu, dass ein großer Teil der privaten und auch der geschäftlich genutzten WLANs nicht oder nur sehr unzureichend gesichert wurden. Erst nachdem mehr und mehr Meldungen über illegale Aktivitäten mittels fremder WLAN Zugänge in der Öffentlichkeit bekannt wurden, reagierten die Hersteller – mit proprietären Lösungen. Bei den Herstellerlösungen zur einfacheren Einrichtung des WLANs werden mit sogenannten Location-Limited- beziehungsweise Time-Limited-Channels gearbeitet.

Werden zum Beispiel Access Point und WLAN-Dongle von AVM verwendet, genügt es, das Dongle vor der ersten Verwendung kurz in den Accesspoint zu stecken um die sichere Verbindung automatisch zu konfigurieren – diesen physische Zugriff beschreibt der Location Limited Channel. Den Time-Limited Channel findet man beispielsweise bei Lösungen von Linksys, hier muss sowohl an Accesspoint als auch am verwendeten Dongle gleichzeitig ein Knopf betätig werden um einen Sicherungskanal automatisch zu konfigurieren.

Natürlich können diese einfachen Lösungen zur Konfiguration nicht mehr verwendet werden, sobald Sender und Empfänger im WLAN von verschiedenen Herstellern stammen.

4.2 Wahl der besten Konfigrationsmöglichkeit automatisieren

Noch komplizierter ist die Entscheidung für ein geeignetes Protokoll bei der Einrichtung einer verschlüsselten Ende-zu-Ende-Verbindung, beispielsweise eines Virtuellen Privaten Netzes (VPN). Solch eine Wahl ist immer ein Kompromiss zwischen der möglichst einfachen Einrichtung auf der einen und der maximalen möglichen Sicherheit auf der anderen Seite.

Beispielsweise ist der Aufbau einer gesicherten Verbindung mittels der Windows eigenen VPN Lösung für den Benutzer durch einen Verbindungsassistenten weniger kompliziert und

[6] Wired Equivalent Privacy - unsicheres Verschlüsselungsverfahren basierend auf dem RC4 Stromchifre Algorythmus.
[7] Wi-Fi Protected Access – Nachfolgeverfahren zu WEP.
[8] Advanced Encryption Standard - Lizenzfreies symetrisches Kryptosystem.

durch den Einsatz von L2TP und IPSec auch sehr sicher. Abseits einer homogenen Microsoft-Umgebung, erfordert das Einrichten einer gesicherteen Verbindung mittels IPSec durch die vielfältigen und komplizierten Konfigurationsmöglichkeiten eine längere Einarbeitungszeit und birgt ein hohes Fehlerpotential.

Hilfreich wäre hier ein Verfahren, dass die verschiedenen Protokolle und Konfigurationen bewertet und automatisch die für die Aufgaben des Anwenders am besten geeignete Möglichkeit auswählt. Die Autoren Lars Völker, Christoph Werle und Martina Zitterbart stellen in ihrer Arbeit „Decision Process for automated Selection of Security Protocols“ [VöWZ08] eine Verfahren vor, mit dem Sicherheitsprotokolle basierend auf der gebotenen Sicherheit, Dienstgüte und Energieeffizienz bewertet werden können.

Der Selektionsprozess gliedert sich dort in vier Schritte:

- Sammeln der verfügbaren Kandidaten.
- Filtern der Kandidaten aufgrund der Anforderungen des Benutzers.
- Evaluieren der Kandidaten.
- Selektion der Kandidaten mit der höchsten Bewertung.

Zur Bewertung der Sicherheit der verfügbaren Kandidaten wird zunächst die gebotene effektive „Bitstärke“ herangezogen. Unter Bitstärke bezeichnet man den vom NIST Computer Security Devision definierten Vergleichswert basierend auf der besten bekannten Angriffsstrategien gegen das Verfahren. Die Autoren bezogen dabei den Schlüsselaustausch, die Authentifizierung, die eigentliche Verschlüsselung und die Authentifizierung der Nachrichten in die Berechnung der Bitstärke mit ein.

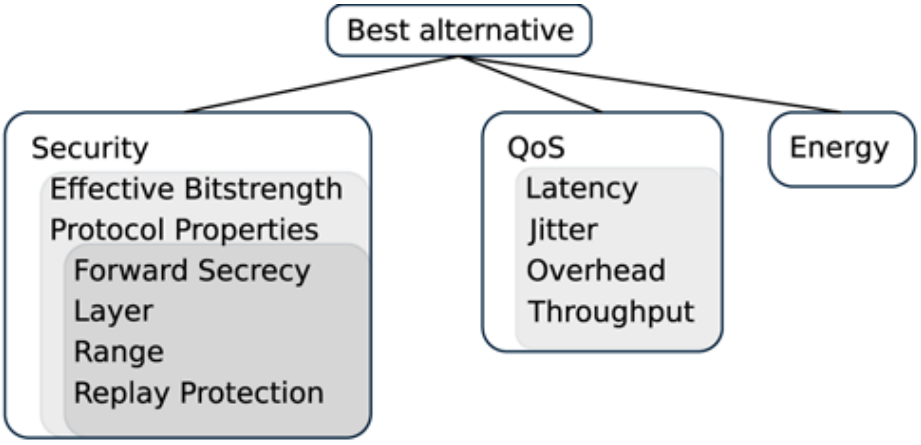

Abbildung 9: Hierarchie der Bewertungskriterien. Quelle: [VöWZ08]

Neben der Bitstärke gingen auch Protokolloption wie Forward Secrecy[9] und die Resistenz gegen Replay-Angriffe[10] in die Bewertung des Protokolls mit ein.

Da der Einsatz einer Verschlüsselung auch die Dienstgüte der Übertragung beeinflusst, werden die Protokolle auch hinsichtlich der entstehenden Verzögerung, des Jitters[11], des maximalen Durchsatzes und des nötigen Overheads bewertet.

[9] Aus einem aufgedeckten Schlüssel kann nicht auf vorhergehende oder nachfolgende Schlüssel geschlossen werden.

[10] Zuvor von einem Angreifer abgefangene Nachrichten, werden vom Empfänger nicht ein zweites Mal akzeptiert.

[11] Schwankungen in der Datenübertragung – hier verursacht durch höheren Rechenaufwand.

Als dritter Parameter geht der Energieverbrauch des Verfahrens in die Bewertung mit ein. Dieser Parameter wird angesichts der steigenden Anzahl an mobilen Geräten in Zukunft wohl eine immer wichtigere Rolle spielen.

Da das beste Protokoll nur hinsichtlich der aktuellen Situation und Aufgabe des Benutzers gewählt werden kann, gehen die einzelnen Bewertungen mit einem, vom Anwender bestimmten, individuellen Gewicht in die Summe ein.

Mit diesem Verfahren und nachfolgender automatischer Konfiguration des gewählten Protokolls, sollte es dem Benutzer möglich sein das für seine Aufgabe bestgeeignete Sicherheitsverfahren auch ohne tiefgreifendes Hintergrundwissen zu wählen.

5 Fazit

In dieser Arbeit wurden einige Sicherheitsaspekte der beliebtesten Anwendungsfelder des Internets, Web und eMail beleuchtet. Am Beispiel des Webbrowsers wurde gezeigt, wie eine fundierte und trotzdem verständliche Rückmeldung and den Benutzer über mögliche Sicherheitsrisiken aussehen kann. Weiterhin wurden verschiedene Richtlinien vorgestellt, deren Einhaltung zu einem verständlicheren Anwendungsdesign für Sicherheitsapplikationen führen kann. Mit PGP, besteht zwar eine kostenlose und sichere Möglichkeit, eMail-Nachrichten zu signieren und verschlüsseln, trotzdem findet diese Möglichkeit in der breiten Bevölkerung durch ihre komplexere Struktur und undurchsichtiger Anwendungen wenig Akzeptanz. Mit dem Verfahren zur automatischen Selektion von Sicherheitsprotokollen wurde eine Möglichkeit vorgestellt, mit der auch unerfahrene Benutzer halbautomatisch über sichere Kanäle kommunizieren können.

Literatur

[Deut07] Statistisches Bundesamt Deutschland. Entwicklung der Informationsgesellschaft, IKT in Deutschland. Webreport, October 2007. www.destatis.de.

[Grou08] Miniwatts Marketing Group. World Internet Users and Population Stats, Jun 2008. http://www.internetworldstats.com/stats.htm.

[Gutm06] Peter Gutmann. Nur für Spezialisten: Benutzbarkeit von Sicherheitssoftware in der Kritik - Teil 1. *Linux Magazin* Band 01/2006, Januar 2006.

[Payn08] W.K. Payne, B.D.; Edwards. A Brief Introduction to Usable Security. Technischer Bericht, Georgia Institute of Technology, Jun 2008.

[Robe05] Roy A. Maxion Robert W. Reede. User Interface Dependability through Goal-Error Prevention. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks*, Washington, DC, USA, 2005. IEEE Computer Society, S. 60–69.

[VöWZ08] Lars Völker, Christoph Werle und Martina Zitterbart. Decision Process for Automated Selection of Security Protocols. In *33nd IEEE Conference on Local Computer Networks (LCN 2008)*, Montreal, QB, Canada, Oktober 2008. IEEE, S. 223–229.

[WhTy99] Alma Whitten und J. D. Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, Berkeley, CA, USA, 1999. USENIX Association, S. 14–29.

[Yee02] Ka Yee. User Interaction Design for Secure Systems. Technischer Bericht, University of California at Berkeley, Berkeley, CA, USA, 2002.

[Yee05] Ka-Ping Yee. Secure Interaction Design. Webreport, Jun 2005. http://zesty.ca/sid.

Abbildungsverzeichnis

RFID-Sicherheit am Beispiel Mifare Classic

Klaus-Martin Scheuer

RFID (**R**adio **F**requency **Id**entification) ist ein Sender-Transponder-System zur Identifizierung von Gegenständen und Lebewesen. Seine steigende Verbreitung ist der geringen benötigten Größe der Transponder und einem niedrigen Fertigungspreis geschuldet. Eine der häufigsten Anwendungen ist das Platzieren des Transponders in Karten, die z.B. zum Ausweisen, zur Zutrittskontrolle oder auch als Geldkarte Einsatz finden. Diese kritischen Bereiche verlangen nach Wahrung von Privatsphäre und Besitz und werfen die Frage nach der Sicherheit dieser Systeme auf. Die kontaktlosen Speicherkarten mit Sicherheitsfunktionen der Mifare-Classic-Familie stellen mit über einer Milliarde Mifare-Karten in Benutzung weltweit den größten Marktanteil. Dies entspricht laut dem Hersteller NXP einer Deckung von 70% des Marktes der kontaktlosen Smartcards [NXPc]. Die Mifare-Classic-Speicherkartenfamilie soll zur Betrachtung der Sicherheitsaspekte von RFID Gegenstand dieser Ausarbeitung sein.

1 Einleitung

RFID-Tags erfahren eine ständig steigende Verbreitung für die Speicherung von schützenswerten Informationen. Hierbei sollte die Sicherheit eine hohe Priorität genießen, jedoch wird diese Anforderung oft zu Gunsten niedrigerer Anschaffungskosten wenig Beachtung zu teil.

NXPs Mifare Classic scheint auf den ersten Blick beides zu bieten: Niedrige Anschaffungskosten und sichere Verschlüsselung durch den hauseigenen Crypto-1-Algorithmus. Weiterhin verfügt sie über eine höhere Übertragungsgeschwindigkeit als die Konkurrenzprodukte. Der Crypto-1-Algorithmus wird zwar von NXP geheimgehalten, jedoch war es möglich durch Mitlesen der Kommunikation zwischen Karte und Lesegerät und physikalischem Reverse-Engineering auf den tatsächlich implementierten Algorithmus zu schließen. Hierfür wurde der Chip abgeschliffen und die einzelnen Chipebenen fotografiert. Aus diesen Fotos ließen sich alle relevanten Gatter erkennen und durch Verfolgung der Leiterbahnen die logischen Verknüpfungen des Algorithmus' erkannt werden.

Eine Folge der Analyse war die Erkenntnis über Schwachstellen des Verschlüsselungsalgorithmus, eine genügende Sicherheit für schützenswerte Informationen kann nicht gewährleistet werden.

2 RFID-Funktionsweise

RFID-Systeme werden zur Identifikation über Funkkanäle verwendet. Ein solches Identifikationssystem besteht immer aus einem Sender und mindestens einem Transponder. Ein typischer Transponder kann als einfacher Tag an Dingen angebracht sein oder in eine Smartcard integriert werden. Weiterhin gibt es Versuche wie Implantate in Menschen, die zur Identifikation

und für medizinische Daten Anwendung finden sollen. Smartcards unterscheiden sich von solchen Tags durch eine eigene integrierte Logik, die zu Berechnungen und zur Authentifizierung genutzt werden kann. Bei der Vielzahl der sich im Umlauf befindlichen RFID-Kartensysteme wird hauptsächlich im 120-135 kHz Niederfrequenz- und im 13,56 MHz Hochfrequenzbereich agiert. Der 13,56-MHz-Bereich ist im ISO-Standard 14443 definiert und soll im folgenden näher betrachtet werden. Im Standard ist die Nachbereichs-Ankopplung (maximal zehn Zentimeter) für Karten und Lesegeräte definiert. Als Fachtermini werden PICC (**P**roximity **I**ntegrated **C**ircuit **C**ard) und PCD (**P**roximity **C**oupling **D**evice) für Karte und Lesegerät verwendet. Die Karte bezieht ihre benötigte Spannung aus dem vom Lesegerät erzeugten Induktionsfeld. Das PCD kommuniziert über eine Modulation auf diesem Feld, die PICC über gesteuertes Ausschalten einer Last (z.B. einem Widerstand). Für alle definierten Kartentypen (A und B) ergibt sich eine Basisdatenrate von etwa 105,9 kbit/s.

2.1 Kartentypen

ISO/IEC 14443 unterscheidet zwei Kartentypen: Typ A und Typ B. Der Hauptunterschied zwischen diesen Typen betrifft die Methoden zur Modulation, die Kodierungsschemata und die Prozeduren zur Protokollinitialisierung. Auf Typ A soll hier näher eingegangen werden.

2.2 ISO 14443 Typ A

Für Typ A wird eine 100% ASK (**A**mplitude **S**hift **K**eying, Amplitudenmodulation) zur Kommunikation vom PCD zum PICC verwendet. Bei dieser Art der Modulation wird zwischen voller Stärke und keinem Trägerfeld umgeschaltet. Durch die fehlende Energieübertragung bei Abschalten des Trägerfeldes muss die Zeit für die Modulation des Feldes gering gehalten werden. Dies wird durch eine modifizierte Millerkodierung erreicht, durch die man Ausschaltzeiten von 2 bis 3 μs erreichen kann. Es wird immer nur etwa eine Viertelbitlänge lang moduliert, dadurch kann es keine direkt aufeinanderfolgende Feldabschaltungen geben. Die Kommunikation in Richtung des Lesegeräts wird mit der oben erwähnten Lastmodulation realisiert. Hier werden die Daten mit der Manchesterkodierung auf einen 847 kHz Subträger moduliert. Jede Karte verfügt über eine universale Identifikationsnummer, der sogenannten UID.

In ISO 14443 wird der Typ A noch weiter in Part 1-4 unterteilt.

2.2.1 Part 1 – Physikalische Eigenschaften

Im Part 1 werden physikalische Eigenschaften der Karte und der benötigten Umgebung definiert. Dies betrifft die Größe der Karte und ihre Widerstandsfähigkeit gegen die Bestrahlung mit UV- und Röntgenstrahlung, sowie ihre Zug- und Knickfestigkeit und den Widerstand gegen elektrische und magnetische Felder. Als Umgebungsparameter wird die Betriebstemperatur definiert.

2.2.2 Part 2 – Funkinterface

In Part 2 wird das Funktinterface definiert, welches zur Daten- und Energieübertragung genutzt wird. Hierbei sind folgende Basiseinheiten vorgegeben:

$f_c = 13,56\,\text{MHz} \pm 7\,\text{kHz}$ (Trägerfrequenz)

$t_{bit} = \frac{128}{f_c} \approx 9,44\,\mu\text{s}$ (Bitdauer)

$f_s = \frac{f_c}{16} \approx 847,5\,\text{kHz}$ (Subträgerfrequenz)

2.2.3 Part 3 – Rahmenformat

In Part 3 ist das Rahmenformat für die Kommunikation, eine Zustandsmaschine und der Kommandosatz für die Antikollision und Selektion der Karten definiert. Dieser Abschnitt ist sehr umfangreich, es soll hier die, für die Ausführungen der Sicherheit später relevante, Antikollision verkürzt erläutert werden.

Ablauf der bitorientierten Antikollision bei Typ A

Der Algorithmus beginnt mit einem REQA- (**Req**uest Type **A**) oder WUPA-Kommando (**W**ake-**up** Type **A**) vom Lesegerät an die erreichbaren Karten. Die Karten antworten mit einem ATQA-Kommando (**A**nswer **T**o Re**q**uest Type **A**). In diesem Kommando sind Informationen über die UID-Länge enthalten. Das Lesegerät hat nun die Bestätigung, dass sich Karten im Feld befinden und sendet nun ein ANTICOLLISION-Kommando, auf dass die Karten mit ihrer UID antworten. Im Lesegerät werden nun die manchesterkodierten UIDs ausgewertet und bestimmt, ob es sich um eine oder mehrere Karten handelt. Diese Unterscheidung wird durch die Manchesterkodierung möglich, da hier nur in einer Bithälfte moduliert wird. Ist ein Signal auf die gesamte Dauer des Bits moduliert, ist dies ein Anzeichen für eine Kollision: An dieser Stelle unterscheiden sich die UIDs von mehreren Karten. Fand keine Kollision statt, spricht das Lesegerät die Karte mit einem SELECT-Kommando und der UID direkt an, alle anderen Karten werden mittels HLTA, dem Haltekommando, schlafen gelegt. Gibt es jedoch eine Kollision wird ein ANTICOLLISION-Kommando mit allen UID-Bits bis direkt vor dem unterschiedlichen Bit gefolgt von einer 0 oder 1 gesendet. Als Folge muss eine Karte weniger antworten. Dies wird solange wiederholt, bis es keine Kollisionen mehr gibt.

2.2.4 Part 4 – Übertragungsprotokoll

In Part 4 wird ein blockorientiertes halbduplex Datenübertragungsprotokoll definiert. Es werden die Verhandlungen über die Baudrate zwischen Karte und Leser, ein Blockformat zur Datenkapselung, die Verkettung zur Aufteilung längerer Blöcke, Fehlerbehandlung und Recovery-Szenarien beschrieben. Um die Anforderungen des ISO-Standards zu erfüllen, ist die Implementation des optionalen vierten Parts nicht nötig. Lediglich Lesegeräte, die mit Karten von mehreren Herstellern kommunizieren, benötigen die Implementation aller vier Parts zur vollen ISO-Kompatibilität.

3 Mifare Classic

1995 stellte NXP, damals noch Philips, die Mifare-Kartenfamilie für die Einsatzgebiete öffentlicher Verkehr, Zugangskontrollsysteme und Event Ticketing vor. Mifare Classic ist der Oberbegriff einer Teilfamilie der Mifare-Kartenreihe. Allen Mitgliedern dieser Familie ist der Crypto-1-Verschlüsselungsalgorithmus zugleich. Sie zeichnen sich durch ein gutes Preis/Leistungsverhältnis aus und werden als kontaktlose Speicherkarten mit Sicherheitsfunktion vertrieben. Der Hauptunterschied der einzelnen Kartentypen stellen die verschiedenen Speichergrößen dar. Beliebt wurden sie durch ihre hohe Datenübertragungsrate von etwa 106 kbit/s. Die Karten sind kompatibel zu ISO 14443 Part 1 bis 3, ihre UID ist fest einkodiert und ist vier Byte lang. Part 4 des ISO-Standards 14443 decken sie nicht ab, es wird ein proprietäres geheimes Protokoll verwendet [NXPa].

3.1 Authentisierung

Die Mifare-Classic-Authentisierung wurde als 3-Wege-Authentisierung (Mutual Authentication) nach dem ISO-Standard 9798-2 realisiert. Dem Standard nach müssen folgende Bedingungen erfüllt sein:

- Der verwendete Zufallszahlengenerator muss als kryptografisch stark anzusehen sein.
- Der verwendete *symmetrische Systemschlüssel* muss vor einem Angreifer geheimgehalten werden.
- Das Verschlüsselungsverfahren muss kryptografisch stark (ungebrochen) sein und mindestens 80 Bit lange Schlüssel verwenden.

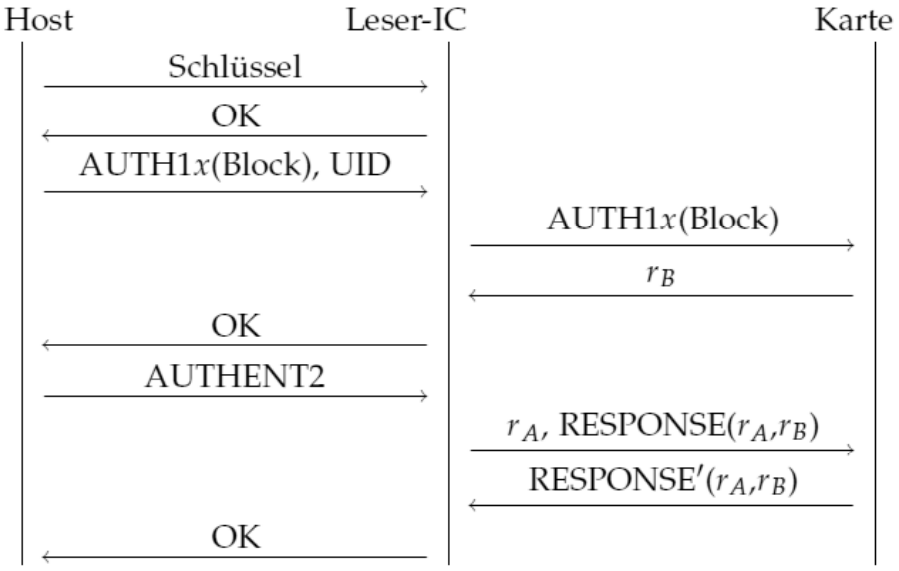

Abbildung 1: Authentisierung mit Schlüssel $x \in \{A, B\}$ [Plöt08]

Die Authentisierung wird vom Host angestoßen, indem dem Leser-Chip der zu verwendende Schlüssel übergeben wird 1. Daraufhin wird AUTH1x-Kommando zum Leser-IC geschickt, wobei x ein A oder B für den entsprechenden Schlüssel ist. Die Parameter dieses Kommandos sind der zu lesende Block und die UID der Karte. Der Leser-Chip sendet dieses Kommando an die Karte, wobei die UID nicht an die Karte gesendet wird. War die 1. Stufe der Authentisierung erfolgreich, antwortet die Karte mit einem zufällig generiertem Nonce (ein pseudozufällig generiertes Wort, das nur kurz verwendet werden soll). Diese Nonce stellt die sogenannte Challenge für den Leser-IC dar. Er muss nun eine korrekte Antwort auf die Challenge senden. Der Host schickt daraufhin das AUTHENT2-Kommando an den Leser-Chip, die zweite Phase der Authentisierung wird eingeleitet. Der Leser-Chip sendet eine eigene Nonce und eine zur Challenge passende Antwort, die sogenannte Response. Die Karte muss nun ihrerseits eine passende Response an den Leser-Chip senden, der diese überprüft. Dieses Verfahren wird Challenge-Response-Authentifizierung genannt [NXPa, Plöt08, Fox08a].

3.2 Zugriffsrechte

Jede Karte enthält einen EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory) Nur-Lese-Speicher, der sich in Sektoren unterteilt. Jeder dieser Sektoren unterteilt sich seinerseits in Blöcke. Bei einer Mifare Classic 1K sind das z.B. 16 Sektoren unterteilt in 4 Blöcke à 16 Byte (64 Blöcke insgesamt). Je Sektor sind zwei verschiedene Schlüssel definiert,

die Zugriffsrechte je Schlüssel können je Block gesetzt werden. Block 0 im ersten Sektor ist permanent schreibgeschützt, er wird Manufacturer-Block genannt und enthält neben herstellerspezifischen Daten die UID der Karte, sowie eine Prüfsumme über die UID. Die Zugriffsrechte auf alle anderen Blöcke hängen von dem Schlüssel ab, der in der Mutual Authentication verwendet wurde. Der letzte Block eines jeden Sektors enthält die Zugriffsschlüssel und -rechte für den kompletten Sektor. Er ist generell nicht für Benutzerdaten zu verwenden und wird als Sektor-Trailer bezeichnet.

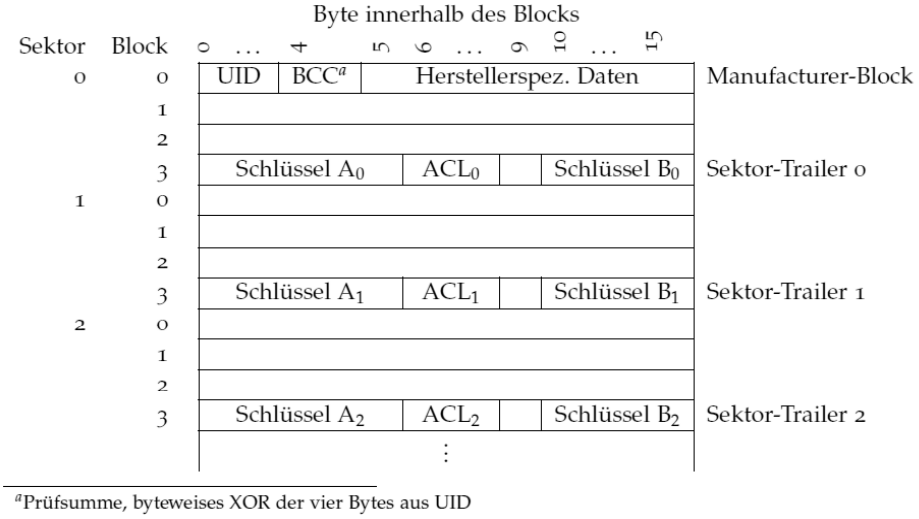

Abbildung 2: Speicherlayout der Mifare-Classic-Karte [Plöt08]

Je Sektor gibt es genau einen Sektor-Trailer. Die Zugriffsrechte sind in ACL-Feldern gespeichert, die sich in der Mitte des Sektor-Trailers befinden. Der Sektor-Trailer folgt dem Format Schlüssel A | ACL-Felder | Schlüssel B. Für jeden Block gibt es ein ACL-Feld, welches aus sechs Bits besteht. Die ersten drei Bits enthalten die Zugriffsrechte, es folgen drei weitere Bits, die die Invertierung der ersten drei darstellen. Aus diesen Bits leiten sich sowohl die Zugriffsrechte für den Sektor-Trailer selbst, als auch auf die restlichen Blöcke ab. Schlüssel A kann nicht ausgelesen werden, Schlüssel B kann nur ausgelesen werden, wenn man sich mit Schlüssel A authentifiziert hat [NXPa].

Ein möglicher Angriff auf Smartcardsysteme stellt die Bestrahlung mit UV-Licht dar. Durch die Bestrahlung lassen sich gezielt Speicherzellen des EEPROM zurücksetzen. Dieser Angriff ist bei Mifare Classic nicht anwendbar, denn bei jedem Speicherzugriff wird das Format der ACL-Felder gecheckt. Wurden diese Speicherzellen mit UV-Licht bestrahlt, ist die Invertierung verloren gegangen. Der Sektor wird bei einem Verstoß gegen dieses Format von der Karte unwiderruflich gesperrt [Plöt08].

Die Konfiguration der Zugriffsrechte erfolgt mehrdimensional. Je Block lassen sich je Schlüssel Zugriffsrechte konfigurieren. Die Rechte für normale Datenblöcke können auf mehrere Arten spezifiziert werden: Zum einen kann sowohl ein Lese-/Schreibzugriff gewährt werden, zum anderen können Datenblöcke auch nur lesbar sein. Als dritte Möglichkeit gibt es die Sperrung des Blockes. Man unterscheidet außerdem reine Datenblöcke von Wertblöcken, die für den Einsatz in Wertkarten konzipiert sind und über weitere Befehle angesprochen werden könnnen. Primär ist das das Inkrementier- als auch das Dekrementierkommando, aber auch das Übertragen des Wertes in einen anderen Block, sowie der Wiederherstellung des Blockes. Das Aufwerten oder Abbuchen muss vom einem Transferkommando gefolgt werden, das den neuen Wert in den Wertblock schreibt. Das Restore-Kommando zur Wiederherstellung kann vor dem Verändern des Wertblocks zur Sicherung seines Wertes verwendet werden. Dazu

wird der Inhalt des Wertblocks in das interne Speicherregister geladen und kann von dort mit einem Transferkommando wieder in den Block zurückgeschrieben werden. Wertblöcke speichern eine 32-Bit-Ganzzahl in einem speziellen Format, das sich aus zwei nicht invertierten und einer invertierten Kopie der Zahl zusammensetzt. Zusätzlich zur Unterscheidung nach Lese-/Schreibrecht wird nach Aufwertrecht und Abbruch-Transfer-Restore-Recht unterschieden [NXPa].

4 Mifarespezifische Schwachstellen

Für die Nonce/Challenge in der Mutual-Authentication-Phase wird eine Zufallszahl benötigt. Dazu verwendet man einen 16-Bit-Pseudozufallsgenerator, der mit einem LFSR (**L**inear **F**eedback **S**hift **R**egister, einem linear rückgekoppeltem Schieberegister [Fox08b]) realisiert wurde. Je Takt wandern alle im Schieberegister enthaltenen Bits um eine Position nach hinten. Die Eigenschaft des linearen Feedbacks besagt, dass das frei gewordene Bit am Registeranfang mit dem Ergebnis der XOR-Verknüpfung einiger höherer Registerbits aufgefüllt wird. Das LFSR hat eine Filterfunktion, die mehrere Registerinhalte logisch verknüpft, um einen Output zu erhalten. In jedem Takt errechnet die Filterfunktion auf diese Weise ein Bit des Schlüsselstroms. Dieses Bit wird später zum Verschlüsseln des Klartextes per XOR-Verknüpfung verwendet. Die entstehenden Zufallszahlen unterliegen einer häufigen Wiederholung, was zu suboptimalen Avalancheeigenschaften (kleine Änderung am Klartext führt zu großer Veränderung des Geheimtextes) führt. Eine weitere aus kryptographischer Sicht wichtige Eigenschaft der Folgenlosigkeit (durch das Wissen über einen Zustand in der Verschlüsselung können keine Aussagen über vorherige oder spätere Zustände getroffen werden) ist nicht gegeben. Durch die fehlende Nichtlinearität des LFSR kann man mit Hilfe eines bekannten Plaintextes jeden Zustand desselben berechnen.

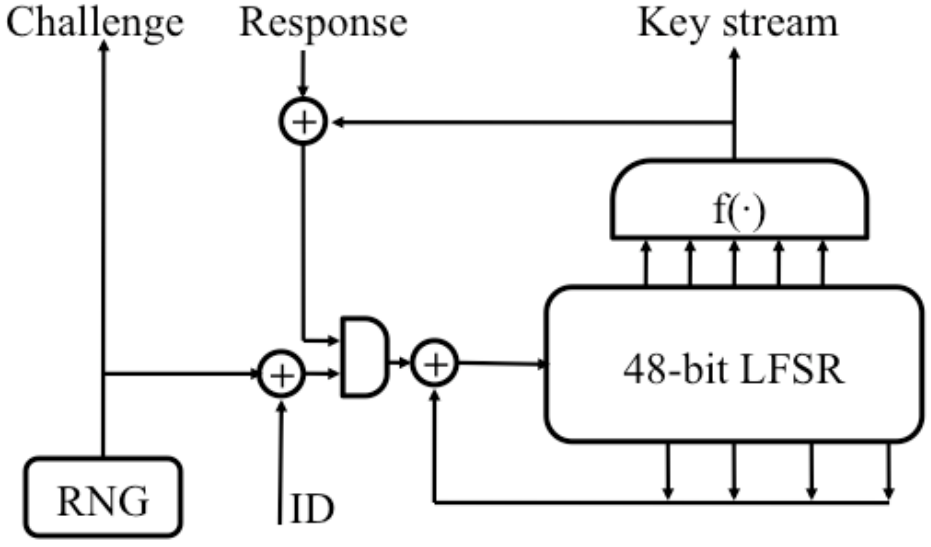

Abbildung 3: Crypto-1 Stromchiffre und Initialisierung [NESP08]

Das LFSR wird mit 105,9 kHz getaktet. Da es nur $2^{16} - 1$ Zustände hat, der Zustand aus Nullen ist nicht gültig, wiederholen sich die Zufallszahlen in einem Intervall von 0,618619 s [Plöt08].

5 Angriffe

Bei den Angriffen auf das Mifare-Classic-Kartensystem lassen sich zwei Hauptgruppen ausmachen: Eine Vielzahl der Angriffe laufen gegen das Funkprotokoll, die restlichen zielen auf die Crypto-1-Verschlüsselung im Speziellen. Die hohe Zahl an Angriffsvarianten lässt auf eine hohe Angreifbarkeit des Systems schließen. Je nach Intention des Angreifers wird eine andere Strategie bevorzugt, wobei mehrere Angriffsvarianten zum gleichen Ziel führen können.

5.1 Funkprotokollangriffe

Die erste Untergruppe der Angriffe ist die derjenigen, die nur auf das Funkprotokoll zielen. Diese Art der Angriffe ist nicht mifarespezifisch, sie können auch auf andere RFID-Systeme unternommen werden.

5.1.1 Brute Force

Lange Zeit war das simple Durchprobieren aller möglichen Schlüssel der einzige bekannte Angriff. Lukas Grunwald hat in seinem Vortrag auf der Black-Hat-Konferenz 2006 einen Angriff mit einer Dauer von 25 ms je Authentisierungsversuch vorgestellt. Es ergibt sich für das Testen aller 2^{48} Schlüssel eine Dauer von 222985 Jahren [Grun06]. Selbst mit den von Henryk Plötz vorgeschlagenen Optimierungen lässt sich allenfalls eine Verbesserung um eine Größenordnung erreichen [Plöt08].

Für den Fall, dass man zwei Challenge/Response-Austausche aufgezeichnet hat, haben Nohl et al. zum Abschätzen der Zeit für einen Brute-Force-Angriff einen Aufbau entwickelt, der aus FPGAs (**F**ield **P**rogrammable **G**ate **A**rray) von Pico Computing besteht. FPGAs sind Integrierte Schaltkreise, die sich programmieren lassen und so beliebige logische Strukturen aufweisen können. FPGAs können im Gegensatz zu normalen CPUs vollkommen parallel arbeiten und so mehrere hundert arithmetische Operationen in der gleichen Zeit durchführen, in der man nur wenige Operationen auf herkömmlichen sequentiell rechnenden CPUs ausführen kann. Aufgrund der Einfachheit der Chiffre aus dem Crypto-1-Algorithmus, konnten 6 Pipelines auf einem einzigen Xilinx Virtex-5 LX50 FPGA zusammengeführt werden. Eine funktionierende Implementierung auf einem Array aus 64 solcher FPGAs benötigte für das Durchtesten aller 2^{48} Schlüssel weniger als 50 Minuten [NESP08].

5.1.2 Man-in-the-Middle

Der bei Mifare Classic eingesetzte Verschlüsselungsalgorithmus Crypto-1 ist eine Stromchiffre, d.h. jedes Bit ist verschlüsselt. Crypto-1 verwendet CRC (**C**yclic **R**edundancy **C**heck, zyklische Redundanzprüfung) zur Fehlererkennung des Stroms [NXPa]. CRC kann jedoch nicht die Integrität der Daten bestätigen. Ein Angreifer kann leicht einen anderen Datenstrom erzeugen, der den gleichen CRC-Wert wie eine regulär verschlüsselte Nachricht hat.

Um einen erfolgreichen Man-in-the-Middle-Angriff durchführen zu können, muss man die Kommunikation verändern können und Teile des Klartextes kennen oder raten können. Hierzu bietet sich ein Relay-Angriff (siehe 5.1.3) an. Der Klartext kann z.B. beim Geldkartenaufladen vom Display des Automaten abgelesen werden. Der Geldwertbetrag wird als 4 Byte-Ganzzahl übermittelt und enthält üblicherweise den Geldbetrag in Cents [dKGHG08].

5.1.3 Relay-Angriffe

Für einen Relay-Angriff verwendet man einen PICC-Emulator und einen PCD. Beide müssen das Mitlesen der übermittelten Daten unterstützen. Der Angreifer hält den PCD an die zu attackierende Karte und benutzt die PICC zum Mitlesen der Kommunikation von der Karte. Hancke beschreibt einen einfachen Aufbau, der einen solchen Relay-Angriff ermöglicht. Man bringt einen sogenannten Proxy (den PICC-Emulator) in der Nähe des Lesegerät und einen sogenannten Mole (den PCD) in der Nähe der Karte an. Somit kann man die Kommunikation der Karte mit dem Lesegerät mitschneiden. Durch Hanckes Aufbau ergeben sich $20\mu s$ Verzögerung, die aber in der Praxis trotz Abweichung vom Standard kein Problem darstellen [HaKu05].

5.1.4 Wiederholungsangriffe

Unter einem Wiederholungsangriff (engl. replay attack) versteht man das Abspielen einer aufgezeichneten Kommunikation. Ein solcher Angriff setzt voraus, dass die Zustände in der Authentisierungsphase zeitabhängig sind, d.h. dass unter Einhaltung der gleichen Timings immer der gleiche verschlüsselte Text erzeugt wird. Der Startzeitpunkt des Abspielens kann gewählt werden, die darauffolgende aufgezeichnete Kommunikation ist statisch, es kann nicht dynamisch auf Veränderungen des Kommunikationsverhaltens eingegangen werden. Somit ist eine Anwendung auf zeitunabhängige Authentisierungen nicht möglich, es fehlt die nötige Dynamik und zusätzliches Wissen über den Kommunikationsvertrag der Authentisierungsphase. Beim pseudozufälligen Crypto-1-Algorithmus ist diese Zeitabhängigkeit gegeben, man benötigt kein weiteres Wissen über den den genauen Aufbau des Crypto-1-Algorithmus'. Um einen Wiederholungsangriff auf eine Mifare-Classic-Karte vorzunehmen, schneidet man zuerst eine geglückte Authentisierung mit, extrahiert die Kartennonce und berechnet dann zu welchem Zeitpunkt nach Aktivieren des Induktions-Feldes man mit der Authentisierung beginnen muss, um die gleiche Nonce von der Karte zu erhalten. Spielt man nun die aufgezeichnete Authentisierung korrekt ab, kann man den Rest der mitgeschnittenen Kommunikation an den PCD schicken [Plöt08].

Für den Angreifer interessante Szenarien:

- Aufwertkommando: Erhöhe Geldbetrag um 10 Euro
- Schreibkommando: Schreibe Wert der U-Bahn-Benutzungen
- Lesekommando: Veränderte Daten können gefunden werden. Diese Daten können für ein Reverse-Engineering-Angriff genutzt werden oder aber stellen einfach ein Privatsphären-Problem als ein Indikator für die Benutzung der Karte dar

5.1.5 Keystream-Recovery

Man schneidet die Kommunikation zwischen PCD und PICC mit und spielt sie einem PCD unter Kontrolle in passender Zeit vor (vgl. 5.1.4), modifiziert den Klartext, damit die Karte ein Kommando, zu dem man den Klartext der Antwort kennt, erhält. Dann berechnet man zu jedem bekannten Segment von bekanntem Klartext den entsprechenden Keystream (Schlüsselstrom). Auf diese Weise ist man in der Lage den Keystream des Mitschnitts teilweise entschlüsseln. Dazu variiert man die Kommandos und/oder Parameter bis man den kompletten Keystream kennt. Der Angriff basiert darauf, dass man aus der Kenntnis von einem von zwei Klartexten P_1, P_2 und den zugehörigen verschlüsselten Daten C_1 und C_2 bei Verschlüsselung mit dem gleichen Keystream K, den anderen Klartext offenlegen kann. Es gilt folgender Zusammenhang:

$$\left.\begin{array}{l} P_1 \bigoplus K = C_1 \\ P_2 \bigoplus K = C_2 \end{array}\right\} C_1 \bigoplus C_2 \Rightarrow P_1 \bigoplus P_2 \bigoplus K \bigoplus K \Rightarrow P_1 \bigoplus P_2$$

[dKGHG08]

De Koning et al. haben einen Angriff vorgestellt, mit dem es möglich ist ohne Kenntnis des Schlüssels auf Sektor 0 zuzugreifen. Sie benutzen hierzu eine aufgezeichnete Transaktion zwischen einer Mifare-Classic-Karte und einem PCD und bekannte Speicherinhalte der Karte, die von NXP veröffentlicht wurden [NXPb]. Für den Angriff greifen sie auf den schreibgeschützten ersten Block zu (vgl. hierzu Abbildung 2). Der erste Block enthält die aus der Antikollisionsprozedur bekannten UID und ein BCC-Byte (**B**it **C**ount **C**heck), welches eine Prüfsumme (byteweises XOR aller 4 UID-Bytes) darstellt. Es sind also 5 Byte bekannt. Die restlichen 11 Bytes im Block sind unbekannte Daten, jedoch lassen sich von diesen 11 die ersten 5 Bytes erraten, da sie bei den meisten Karten gleich und daher als bekannt anzusehen sind. Diese bekannten Herstellerdaten (Manufacturer Data) stehen an der gleichen Position innerhalb des Blocks 0 wie die ACL-Felder im Sektor-Trailer. Man kann somit durch abwechselndes Lesen des Blocks 0 und des Sektor-Trailers die ACL-Bytes entschlüsseln. Schlüssel A im Sektor-Trailer ist niemals lesbar, greift man lesend auf ihn zu, enthält der Klartext des Schlüsselstroms 6 Bytes aus Nullen. Anhand der 3 ACL-Bytes lässt sich ablesen, ob der Schlüssel B lesbar ist. In vielen Fällen ist er dies nicht, wie z.B. in der niederländischen Fahrkarte OV-Chipkaart. Ist er nicht lesbar, erhält man entsprechend 6 Bytes an Nullen, greift man lesend auf ihn zu. Nun hat man genügend Informationen um den kompletten Sektor-Trailer zu entschlüsseln und folglich kann man auch den Block 0 entschlüsseln [dKGHG08].

Um höhere Sektoren zu lesen, muss man ähnlich vorgehen, jedoch hat man hier nicht den Vorteil, dass man über die Herstellerdaten Klartext wissen kann. Durch Schlüssel A kann man immer die ersten 6 Bytes bestimmen, meistens auch die letzten 6 Bytes, da wie schon erwähnt Schlüssel B oftmals nicht lesbar ist. Es bleibt ein unbekannter Teil von 4 Bytes Größe: 3 Bytes ACL-Felder und ein weiteres unbekanntes Byte, das oftmals 00 oder 69 ist. Diese fehlenden Bytes in der Mitte zwischen den beiden Schlüsseln müssen nun geschickt erraten werden, z.B. mit Hilfe eines Wertblocks: Kennt man den Betrag im Wertblock, kann man den Klartext von 12 Bytes bestimmen. Im Wertblock werden die 4 Bytes Ganzzahl des Wertes, normal, invertiert und wieder normal gespeichert. Hat man keinen solchen bekannten Wertblock, muss man zu anderen Mitteln wie z.B. Brute Force greifen [dKGHG08].

5.2 Angriffe auf die Crypto-1-Verschlüsselung

Die zweite Untergruppe der Angriffe besteht aus denen, die direkt auf den Verschlüsselungsalgorithmus abzielen. Das Herausfinden des zur Verschlüsselung verwendeten Algorithmus' stellt das gemeinsame Grundprinzip dieser Angriffe dar. Um genügend Kommunikationsdaten zu sammeln, benötigen Angriffe auf die Authentifizierung und die algebraischen Angriffe meistens den vorherigen Angriff auf das Funkprotokoll. Der Hardware-Angriff ist freier, da er keine Vorarbeit benötigt.

5.2.1 Manipulation der Authentifizierungsphase

Um die Kommunikation in in der Authentifizierungsphase unter seiner Gewalt zu haben, muss der Angreifer die Zeiten der Kommunikationspartner steuern können. Dies kann mit Emulatoren für Karten und Lesegeräte durchgeführt werden. Im Aufbau von Nohl et al. wurde OpenPICC als Kartenemulator und OpenPCD als Lesegerätsemulator verwendet. Die Kommunikationsanteile der PICC und die des PCD können mit einem Logikanalysator aufgezeichnet werden. Der Aufbau sieht vor, dass man die anzugreifende Karte zwischen OpenPICC und

OpenPCD legt. Der OpenPICC wird hierzu als Amplitudendemodulator und der OpenPCD sowohl als Subcarrierdemodulator als auch als normales Lesegerät eingesetzt. Eine Erkenntnis aus dem Mitschneiden aus diesem Aufbau ist, dass sich Karten- und Lesegerät-Challenges wiederholen [NESP08].

In der Authentisierungsphase werden UID und Schlüssel durch XOR verknüpft. Im Zuge des Experiments ergab sich eine komplette Zuordnung jedes UID-Bits zum entsprechenden Schlüsselbit. Diese Zuordnung resultiert aus der Rückkopplung des Schieberegisters, ein entsprechender Algorithmus wurde durch Henryk Plötz iterativ bestimmt [Plöt08]. Es lässt sich so zu jeder UID der passende Schlüssel bestimmen, jedoch ist mit einer Größe von einem Petabyte die Tabelle für einen mobilen Einsatz zu groß. Durch den Einsatz einer Rainbow-Tabelle lässt sich bei einer Kettenlänge von 512 Schritten die Größe auf einem Terabyte verringern. So lässt sich eine innerhalb weniger Sekunden ein passender Schlüssel finden [KrNP08].

Garcia et al. konnten eine weitere Schwachstelle zum Angriff auf die Authentifizierung finden. Alle 8 Bits sendet das Kommunikationsprotokoll ein Paritätsbit. Dieses Paritätsbit wird nicht aus dem verschlüsselten Text berechnet, sondern über den Klartext. Diese Paritätsbits selbst werden wieder mit dem nächsten Bit verschlüsselt, das auch zur Verschlüsselung des nächsten Klartextbits verwendet wird. Durch diese Lücke kann man ein Bit an Information über den Klartext für jedes gesendete Byte bekommen. Aus einer generierten Tag-Nonce kann man so die Bits, die mit dem gleichen Keystreambit wie die Paritätsbits verschlüsselt werden, gewinnen. Weiterhin kann man nun einfach feststellen, ob die Paritätsbits gleich zum ersten Bit des nachfolgenden Bytes sind. Dadurch lassen sich die zu testenden Nonces von 2^{16} auf 64 reduzieren. Durch die Vorhersagbarkeit des LFSR kann man mit sehr hoher Wahrscheinlichkeit die nächste Nonce bestimmen. Sie fanden experimentell $d(n_T, n'_T) = 8t - 55c - 400$ als Distanz in Bitperioden (der Zeit, die es dauert ein Bit zu senden, etwa 9,44μs) zwischen zwei Nonces n_T und n'_T, wobei t die Zeit zwischen dem Senden der verschlüsselten Leser-Nonce aus der ersten Authentifizierungssession und dem Authentifizierungskommando, das die nächste Session startet, ist und c die Anzahl der Kommandos, die der Leser in der ersten Session gesendet hat, darstellt. Garcia et al. können jedoch keine Erklärung für diese Relation liefern, sie konnten sie nur experimentell belegen [GKGMR+08].

5.2.2 Algebraischer Angriff

Courtois, Nohl und O'Neil haben einen algebraische Angriff auf Crypto 1 durchgeführt. Sie stellten aus Schlüsselstrombits, Schlüsselbits, Ausgabebits und einer großen Anzahl an Zwischenvariablen ein Gleichungssystem auf und ließen es mit einem SAT-Solver lösen. Experimente ergaben, dass 50 bekannte Schlüsselstrombits genügen um einen Schlüssel zu berechnen. Diese lassen sich z.B. durch die Keystream-Recovery erlangen. Ein Standard-PC löst ein derartiges Gleichungssystem in etwa 200 Sekunden. Mit 4 ausgewählten Initialisierungsvektoren (Karten-Nonces) sind es sogar nur 12 Sekunden. Ein einmaliges passives Mitschneiden einer regulären Nutzung einer unmanipulierten Karte genügt zur wiederholten Authentifizierung. Der ganze Vorgang von Mitschneiden der Kommunikation bis zur fertigen Herstellung eines Kartenklons ist nach Aussage von Courtois et al. in wenigen Minuten durchführbar [CoNO08].

5.2.3 Hardware Reverse Engineering

Courtois et al. haben auch eine physische Analyse des Algorithmus vorgenommen. Die Chips auf der Mifare-Classic-Karte ist in sogenannten ID1-Kartenkörpern vergossen, er lässt sich jedoch in etwa 30 Minuten mit Hilfe von Azeton von der Plastikkarte lösen. Die Analyse des Chips konnte mit einem optischen Mikroskop mit 500-facher Vergrößerung vorgenommen werden. Das zu untersuchende Silizium ist nur einen Quadratmillimeter groß, wenige

zehntel Millimeter dick und besteht aus fünf Chiplagen, die man einzeln mit einem Spezialschleifgerät abtragen muss. Die oberste Ebene des Chips dient nur als Sichtschutz. Zum Abschleifen muss der Siliziumblock mit dem Silizium nach unten und parallel zur Polierfläche auf einen Polierblock geklebt werden. Man poliert mit einem Polierautomaten mit einer Polieremulsion von 0,04 Mikrometern Korngröße. Hierbei muss man sehr vorsichtig vorgehen und oft unter dem Mikroskop den Fortgang kontrollieren. Schräglagen von wenigen Mikrometern machen den Aufbau unbrauchbar, da damit schnell untere Schichten angeschliffen werden. Dies lässt sich vereinfachen durch das Einbetten des Siliziumblocks in einen größeren Plastikblock. Nach dem Abschleifen einer Chipebene, legt man nun den Chip unter das Mikroskop mit einer eingebauten Digitalkamera lassen sich Fotos der Oberfläche der darunterliegenden Ebene schießen. Im Aufbau hat sich gezeigt, dass sich die Fotos am Rechner mittels einer gängigen Panoramabildsoftware zusammensetzen ließen. Nach einer Kantenschärfung lassen sich nun die aus Transistoren zusammengesetzte Gatter erkennen. Aus den tausenden Gattern kann man 70 verschiedene Typen ableiten. Um den Teil der Schaltung zu finden, der für die Kryptografie zuständig ist, muss man nach schwach verbundenen Einheiten suchen, da die Kryptografielogik keinen Einfluss auf die umfangreichere Steuerelektronik hat. Außerdem sind XOR-Gatter für Kryptografielogik charakteristisch. Ein weiterer Indikator auf die Verschlüsselungslogik sind Flipflops, die für das Schieberegister benötigt werden. Um die Schaltkreise zu reproduzieren, muss man die Leiterbahnen durch die verschiedenen Ebenen verfolgen [CoNO08, Plöt08]. Das Ablesen des Algorithmus wird bei den Mifare-Chips nicht durch heute übliche Obfuscation-Techniken erschwert [Fox08a].

Eine wichtige Erkenntnis aus dem Hardware Reverse Engineering zum Angreifen des Algorithmus' ist, dass die Kryptografielogik nicht genügend Eingänge für UID und dem geheimen Schlüssel um beides bei der Initialisierung der Mutual Authentication zu verwenden [Plöt08].

6 Angriffsszenarien

Verschiedene Einsatzzwecke der Mifare-Classic-Karten erzeugen verschiedene Angriffszenarien. Es gibt zum einen die monetär motivierten Fälle bei Debit-Karten und Transportsystemkarten: Die als Zahlungsmittel verwendeten Mifare-Classic-Karten lassen sich wiederholt unberechtigt aufladen, Transportsysteme lassen sich in der Art manipulieren, dass ein Angreifer einen den Wert der Benutzungen zu einem älteren Stand zurücksetzt. Zum anderen gibt es den sicherheitskritischen Bereich der Gebäudezugänge: Hier kann ein Angreifer durch das Auslesen einer Karte oder dem Mitschneiden einer Benutzung genügend Daten sammeln, um die Karte zu klonen und damit Zugang zu geschützten Bereichen erlangen. Ein weiterer, nicht ganz so bedeutender, Fall ist bei der Zeiterfassung von Arbeitszeiten zu beachten. Hat ein Angreifer eine geklonte Karte oder ist er in der Lage die Zeitbuchungen von einem Kartenemulator abzuspielen, kann er Fehlbuchungen im System erzeugen.

7 Schutzmaßnahmen

Bei den Angriffen auf Geldsysteme ist die Gegenmaßnahme naheliegend: Schattenkonten. Der Geldbetrag darf nicht nur auf der Karte, sondern muss auch im Gerät oder einem Netzwerk gespeichert werden. Ein Angriff auf die Karte wird zwar in seiner Gefährlichkeit geschwächt, der Umbau der Infrastruktur erzeugt aber immense Kosten durch Vernetzung der einzelnen Automaten. Die Umstellung auf ein sicheres System kann hier die bessere und günstigere Alternative darstellen. Für Transportsystemkarten ist eine ähnliche Vorgehensweise wie bei den Geldkarten anzuraten, wobei eine Vernetzung von jedem Ticketautomaten, sowohl die außerhalb der eigentlichen Transportmittel, als auch die innerhalb der Transportmittel (z.B.

in Bussen, Straßenbahnen, Zügen etc.), noch finanziell aufwändiger erscheint. Für den Schutz des Gebäudezugangs ist ein wirkungsvolles Mittel die Kombination der Karten mit PIN-Pads, die erst nach der Eingabe einer gültigen PIN (**P**ersönliche **I**dentifikations**n**ummer) den Zugang freigeben.. Hier hilft es dem Angreifer nicht, die Karte nur zur klonen, er muss zusätzlich. durch andere Techniken wie Social Engineering oder der Benutzung von verstecken Kameras, die PIN in Erfahrung bringen.

Literatur

[CoNO08] Nicolas T. Courtois, Karsten Nohl und Sean O'Neil. Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards. Cryptology ePrint Archive, Report 2008/166, 2008.

[dKGHG08] Gerhard de Koning Gans, Jaap-Henk Hoepmann und Flavio D. Garcia. A Practical Attack on the MIFARE Classic, 2008.

[Fox08a] Dirk Fox. Die Mifare-Attacke. *DuD Datenschutz und Datensicherheit* Band 5, 2008, S. 348–350. Artikel.

[Fox08b] Dirk Fox. Linear Rückgekoppelte Schieberegister. *DuD Datenschutz und Datensicherheit* Band 5, 2008, S. 351. Artikel.

[GKGMR+08] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur und Bart Jacobs. Dismantling MIFACE Classic, 2008.

[Grun06] Lukas Grunwald. New Attacks against RFID-Systems. In *Black Hat Briefings*, 2006.

[HaKu05] Gerhard P. Hancke und Markus G. Kuhn. An RFID Distance Bounding Protocol. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, Washington, DC, USA, 2005. IEEE Computer Society, S. 67–73.

[KrNP08] Jan Krissler, Karsten Nohl und Hernyk Plötz. Chiptease - Verschlüsselung eines führenden Bezahlkartensystems geknackt. *c't - Magazin für Computertechnik* Band 8, 2008, S. 80–85. Artikel.

[NESP08] Karsten Nohl, David Evans, Starbug und Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, 2008.

[NXPa] NXP. *MF1ICS50 - Functional specification Mifare Classic 1K* *http://www.nxp.com/acrobat/other/identification/M001053_MF1ICS50_rev5_3.pdf*. NXP Semiconducturs Austria GmbH, 5.3. Auflage. Datenblatt.

[NXPb] NXP. *MF1ICS70 - Functional specification Mifare Classic 4K* *http://www.nxp.com/acrobat/other/identification/M043541_MF1ICS70_Fspec_rev4_1.pdf*. NXP Semiconducturs Austria GmbH, 4.1. Auflage. Datenblatt.

[NXPc] NXP. MIFARE Classic from NXP Semiconductors `http://www.nxp.com/#/pip/pip=[pfp=41863]|pp=[t=pfp,i=41863]`. Website vom 23.11.2008.

[Plöt08] Henryk Plötz. Mifare Classic - Eine Analyse der Implementierung. Diplomarbeit, Humboldt-Universität zu Berlin, 2008.

Abbildungsverzeichnis

Recent Flaws in DNS and Remedies

Heike Hennig

In vielen Artikeln wurde über Cache Poisoning berichtet seit der Sicherheitsexperte Dan Kaminsky Anfang Juli eröffnete, er hätte eine Schwachstelle in einem der Pfeiler des Internets, dem Domain Name System, entdeckt. Da das Domain Name System für die Auflösung eines Domain-Namen in eine IP-Adresse zuständig ist, kann eine gezielte Ausnutzung dieser Schwachstelle dazu führen, dass Angreifer Internetbenutzer auf gefälschte Webseiten umleiten und dort deren private Daten ausspionieren können. Durch diese Bedrohung ist das Interesse an Sicherheitsstandards für das Domain Name System, wie z.B. DNSSEC, gestiegen. Im Rahmen dieser Arbeit soll ein Überblick über verschiedene Angriffsmöglichkeiten und entsprechende Gegenmaßnahmen gegeben werden.

1 Einleitung und Motivation

Stellen Sie sich folgendes Szenario vor. Sie wollen sich im Internet ein Buch bestellen. Sie geben in ihren Browser die Internetadresse des Onlineanbieters Amazon ein und werden auf dessen Webseite geleitet. Sie suchen sich ihren Artikel heraus und gehen virtuell zur Kasse. Nach der Eingabe ihrer E-Mail Adresse und des Passwortes gelangen sie zur Kaufabwicklung. Sie geben ihre Kontodaten oder Kreditkartendaten an und klicken auf bezahlen. Sofort erscheint der Schriftzug „Vielen Dank für Ihre Bestellung“. Die Kaufabwicklung hat funktioniert, jetzt müssen Sie nur noch auf Ihr Päckchen warten, das ein paar Tage später kommt. Doch als Sie einige Wochen später auf Ihren Kontoauszug bzw. Ihre Kreditkartenabrechnung schauen, kommt der große Schock. Es wurde wesentlich mehr abgebucht als erwartet, eventuell sogar das ganze Konto leer geräumt. Was ist da passiert? So oder so ähnlich könnte es ablaufen, wenn Sie Opfer eines Cache Poisoning Angriffs werden.

Bei einem Cache Poisoning Angriff empfängt ein Nameserver oder Client Nachrichten, deren Absender nicht identifiziert werden kann. Der Absender ist ein anderer Nameserver, der von dem Angreifer kontrolliert wird. Die erhaltenen Nachrichten enthalten falsche Daten, die von dem Nameserver bzw. Client in seinen Cache übernommen werden. Fragt nun ein Internetbenutzer, bei diesem Nameserver, nach einer Adresse, so kann er, ohne sein Wissen, auf eine gefälschte Webseite geleitet werden. Da diese gefälschten Seiten oftmals identisch mit den Orginalwebseiten sind, wird dies nicht bemerkt. Der Benutzer gibt persönliche Daten ein, wie z.B. Zugangsdaten oder Kreditkartennummer, die daraufhin, von der gefälschten Webseite, gespeichert werden. Der Angreifer kann sich dann diese Daten zunutze machen.

Am Anfang des Jahres entdeckte der Sicherheitsexperte Dan Kaminsky im Domain Name System eine Schwachstelle, über die sich ein Cache Poisoning Angriff, mit guter Aussicht auf Erfolg, durchführen lässt. Er erkannte die Gefahr, die davon ausging und erarbeitete, zusammen mit namhaften Softwareherstellern, ein Patch, das diese Sicherheitslücke schließen und solche Angriffe erschweren sollte. Dieses Patch wurde dann am 8. Juli 2008 veröffentlicht. Kaminsky gab zu diesem Zeitpunkt noch keine Details, zu der von ihm entdeckten Angriffsmöglichkeit, heraus. Er wollte damit bis zur Blackhat Konferenz, die einen Monat

später stattfand, warten, um den Netzwerkadministratoren Zeit zu geben, ihre Systeme zu sichern. Doch bereits Ende Juli wurden die Details bekannt. Eine Sicherheitsfirma hatte sie versehentlich veröffentlicht. Kurz darauf erfolgten die ersten Angriffe.

Die Methode mit der die, von Dan Kaminsky, entdeckte Sicherheitslücke ausgenutzt werden kann, soll in dieser Arbeit erläutert werden. Außerdem werden noch weitere Angriffsmöglichkeiten vorgestellt und auf Schutzmaßnahmen zur Sicherung des Systems vor solchen Angriffen eingegangen. Doch zuerst sollen mit einem Kapitel über das Domain Name System die benötigten Grundlagen vermittelt werden.

2 Das Domain Name System (DNS)

2.1 Was ist das DNS?

Das Domain Name System, kurz DNS, ist eine verteilte Datenbank, die im Internet für die Namensauflösung zuständig ist. Wenn Computer in Netzwerken miteinander kommunizieren, verwenden sie IP-Adressen. Diese sind je nach Art des verwendeten Internetprotokolls 32-Bit (IPv4) oder 128-Bit (IPv6) lang. Einem Menschen würde es schwer fallen sich so eine IP-Adresse zu merken und in den Computer einzugeben, wenn er eine bestimmte Internetseite besuchen will. Deshalb werden alphanumerische Adressen, so genannte Domainnamen, benutzt. Damit nun bei Eingabe eines Domainnamens in den Webbrowser die entsprechende Internetseite angezeigt werden kann, muss dieser in eine IP-Adresse umgesetzt werden. Diese Aufgabe übernimmt das Domain Name System.

2.2 Geschichte des DNS

Das heutige Internet hat seinen Ursprung im ARPAnet. Dieses Netzwerk wurde Ende der sechziger Jahre vom amerikanischen Verteidigungsministerium realisiert, um wichtige Forschungseinrichtungen der USA miteinander zu verbinden. Zu Beginn des ARPAnet wurden die Rechner noch mit ihrer IP-Adresse angesprochen, doch schon Anfang der siebziger Jahre wurden Hostnamen eingeführt, die als Eingabe anstelle der IP-Adressen traten. Eine einzige Datei namens HOSTS.TXT enthielt alle Informationen, die man über diese Hosts haben musste: eine Namen-auf-Adressen-Abbildung aller im ARPAnet eingebundenen Hosts [Liu02]. Ursprünglich wurde die Datei von jedem Beteiligten mehr oder weniger selbst verwaltet, was zu vielen Kopien und Fehlern führte [KöKu07]. Dadurch wurde die Datei zunehmend inkonsistent. Deshalb übernahm das Network Information Center (NIC) im Jahr 1974 die zentrale Verwaltung. Nun konnten Administratoren Änderungen per E-Mail an das NIC schicken und sich die aktualisierte Datei von deren Host herunterladen. Mit zunehmender Größe des ARPAnet entstanden jedoch einige Probleme. Die Host-Datei wuchs proportional zur Menge der existierenden Hosts. Die Konsistenz konnte nicht mehr sichergestellt werden und bei der Vergabe neuer Host-Namen kam es zu Namenskonflikten, da das NIC zwar IP-Adressen eindeutig zuweisen konnte, jedoch nicht Hostnamen. Außerdem nahm der Datenverkehr so stark zu, dass der Host des NICs die Last nicht mehr alleine tragen konnte. Nur ein neues System für die Verwaltung der Hostnamen konnte diese Schwierigkeiten beheben. Das System sollte die lokale Administration der Daten erlauben und diese gleichzeitig allgemein verfügbar machen [Liu02]. Durch die Verteilung auf verschiedene Hosts konnte die Last, die ein einzelner Host zu tragen hatte, gesenkt werden und man konnte aufgrund der lokalen Verwaltung sicherstellen, dass aktuelle Daten vorliegen (Konsistenz). Damit nun auch noch die Eindeutigkeit der Namen gewährleistet werden konnte, beschloss man einen hierarchischen Namensraum zu verwenden. Im Jahr 1984 veröffentlichte Paul Mockapetris die RFCs (Request for Comments) 882 [Mock83a] und 883 [Mock83b], die eine Beschreibung des Domain Name Systems

enthielten [Liu02]. Drei Jahre später (1987) wurden zwei neue RFCs (1034 [Mock87a] und 1035 [Mock87b]) herausgegeben, die die alten RFCs ablösten [KöKu07]. Sie beschrieben das DNS in der Spezifikation, wie es heute noch benutzt wird. Obwohl sie in der Zwischenzeit durch die RFCs 2065 [EaKa97], 2136 [VTRB97] und 2137 [East97], um sicherheitsrelevante Aspekte, ergänzt worden sind, bleiben die beiden RFCs dennoch die Grundlage für die im Internet verwendete Namensauflösung.

2.3 Aufbau des DNS

Anhand von Abbildung 1 wird im Folgenden der DNS Namensraum beschrieben. Das heutige Domain Name System basiert auf einer Baumstruktur. Der oberste Knoten ist die Wurzel und wird im DNS als Root oder Root-Domain bezeichnet und ist mit dem Null-Label („ “) gekennzeichnet. Ein Label ist der Name eines Knoten, der ihn von den anderen Knoten im Baum bzw. Teilbaum unterscheidet. Ein Teilbaum wird im DNS üblicherweise als Domain bezeichnet. Die Kindknoten der Root sind mit Labels für die Top-Level-Domains, wie z.B. com, net, org oder de, benannt. Will man den Domain-Namen einer Domain bestimmen, so muss man, ausgehend von deren Wurzel, alle Labels der Knoten bis zum Root-Knoten aneinander reihen und diese durch Punkte trennen.

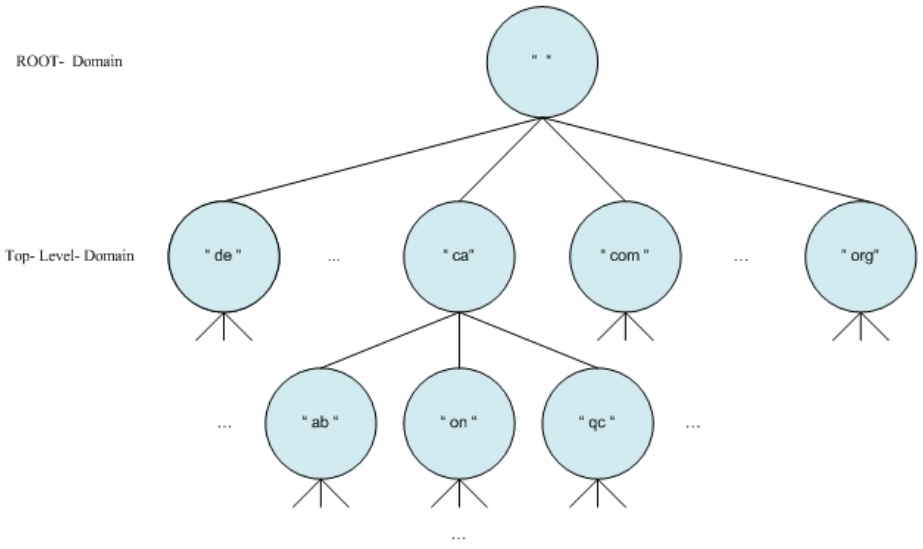

Abbildung 1: DNS Namensraum

2.3.1 Delegierung

Eine Domain kann in verschiedene Subdomains unterteilt werden. Braucht man nicht die gesamte Domain, sondern nur einen Teil davon, so können ein oder mehrere Subdomains an eine andere Organisation delegiert werden. Damit geht die Verantwortung für diese Subdomain an die Organisation über. Sie ist berechtigt die Daten beliebig zu ändern und die Subdomains weiter zu unterteilen und zu delegieren. Die einzige Verbindung zwischen der Domain und einer ihrer Subdomains ist ein Verweis, der zu dem Nameserver der Subdomain führt, damit dieser bei einer Anfrage gefunden werden kann. Dieses Verfahren wird Delegierung genannt.

2.3.2 Zonen

Durch die Delegierung von Subdomains entstehen einzelne Zonen. Eine Zone ist ein kleiner, von der Parent-Domain unabhängiger Teil des Domain-Namensraumes, der eigenständig verwaltet wird. Die Einrichtung von Zonen vereinfacht die Administration. Um den Unterschied, aber auch den Zusammenhang von Domain und Zone zu verdeutlichen, eignet sich am besten ein Beispiel aus dem Buch „DNS und BIND“ [Liu02]. Die Top-Level-Domain ca (für Kanada) hat die Subdomains ab.ca, on.ca und qc.ca für die Provinzen Alberta, Ontario und Quebec. Die Autorität für die Subdomains ab.ca, on.ca und qc.ca könnte an Nameserver in den jeweiligen Provinzen delegiert werden. Die Domain ca enthält alle Daten von ca sowie alle Daten von ab.ca, on.ca und qc.ca. Die Zone ca dagegen enthält nur die Daten von ca, bei denen es sich hauptsächlich um Verweise auf die delegierten Subdomains handelt. Die Zonen ab.ca, on.ca und qc.ca sind von der Zone ca getrennt. Durch die Delegierung besitzt die Zone nur Daten, die für ihren Bereich wichtig ist, während die Domain wesentlich mehr Informationen besitzt, Daten über die Zonen eingeschlossen. Um eindeutig festlegen zu können, welche Nameserver für welches Gebiet zuständig sind, arbeiten Nameserver mit Zonen.

2.3.3 Nameserver und Resolver

Die Programme, die Informationen über den Domain-Namensraum speichern, werden als Nameserver bezeichnet [Liu02]. Jeder Nameserver besitzt Informationen über die Zone für die er zuständig ist. Es ist üblich, dass immer mehrere Nameserver in einer Zone arbeiten. Diese Nameserver werden dabei in zwei Arten unterschieden, die Primary (Master) und Secondary (Slave) Nameserver. Diese Bezeichnungen sagen nichts über die Autorität der Nameserver aus, sondern dienen der Administrationsvereinfachung. Der Unterschied der beiden Serverarten liegt in der Beschaffung ihrer Zonendaten. Master-Nameserver lesen diese aus den so genannten Zonendateien aus, die lokal abgelegt sind. Diese enthalten die Resource Records, die die Zone beschreiben und die Delegierung von Subdomains kennzeichnen. Slave-Nameserver führen dagegen einen Zonentransfer durch, indem sie sich die Daten von dem Master kopieren. Anschließend speichern sie diese Daten in eigene lokale Dateien.

Resolver sind die Clients, die auf Nameserver zugreifen [Liu02]. Auch hier gibt es zwei Arten, Resolvers mit Cache und Stub-Resolvers. Beide Arten können Anfragen an einen Nameserver stellen und dessen Antwort interpretieren. Doch während der Resolver mit Cache seine bereits gesammelten Daten ablegen und bei Bedarf darauf zurückgreifen kann, muss der Stub-Resolver jedes mal erneut eine Anfrage stellen.

In Abbildung 2 wird der Zusammenhang der einzelnen Komponenten verdeutlicht. Im oberen linken Teil der Graphik befindet sich zentriert der Master-Nameserver. Dieser bezieht seine Zonendaten aus der Zonendatei und wird über dynamische Updates aktualisiert. Slave-Nameserver bekommen diese Zonendaten durch Zonentransfers vom Master. Sowohl Master als auch Slave fragen bei dem Caching Forwarder an, wenn ihnen eine IP-Adresse nicht bekannt ist. Der Caching Forwarder kann eine ihm bekannte IP-Adresse direkt an den anfragenden Resolver zurückgeben. Bekommt er eine Anfrage nach einer unbekannten Adresse, so muss er diese bei einem anderen Nameserver (DNS Server 2) erfragen.

2.4 Namensauflösung

Die einzelnen Komponenten des Domain Name Systems wurden bereits ausführlich erläutert. Im folgenden Abschnitt soll nun der Ablauf einer DNS-Anfrage erläutert werden.

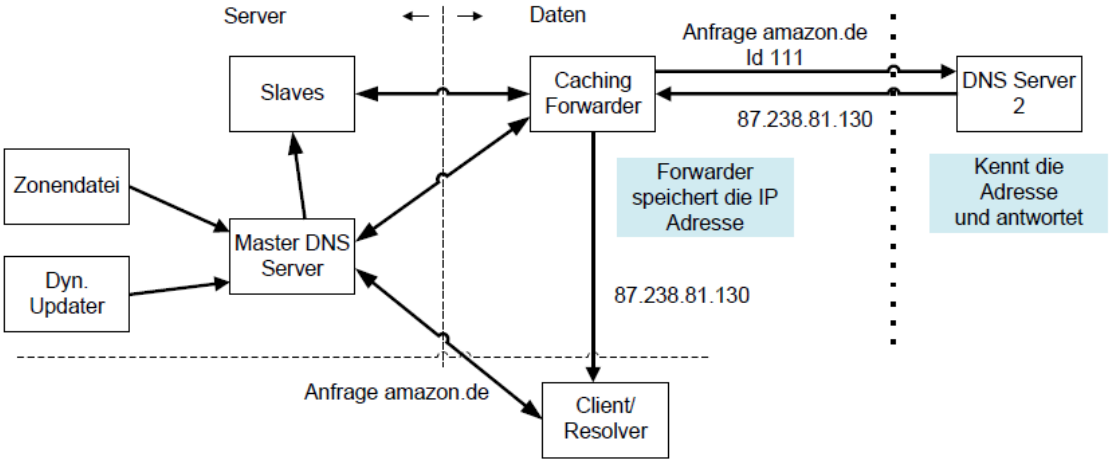

Abbildung 2: Zusammenhang der Komponenten

2.4.1 Prinzip der Namensauflösung

Unter dem Begriff Namensauflösung versteht man den Vorgang, der einen Domain-Namen in eine IP-Adresse umsetzt. Diese Aufgabe übernehmen die Nameserver. Damit ein Nameserver eine Domain erfolgreich auflösen kann, muss er, außer dem Domain-Namen der aufgelöst werden soll, noch die Adressen der Root-Nameserver wissen. So kann jede Adresse aufgelöst werden, indem die Anfrage zuerst an den Root-Nameserver gestellt wird und dieser auf einen Top-Level-Nameserver verweist, der wiederum einen Nameserver nennen kann der noch näher an der gesuchten Domain liegt.

Fragt ein Resolver einen Nameserver nach einer, zu einer bestimmten Domain gehörenden, IP-Adresse, so kann es sein, dass der Nameserver die IP-Adresse bereits kennt und sie an den Resolver zurückgeben kann. Ist die IP-Adresse nicht bekannt, gibt es zwei Möglichkeiten, entweder er gibt die Adresse eines Nameservers, in dem entsprechenden Domainbereich, zurück oder er bringt die IP-Adresse in Erfahrung und gibt sie dann zurück. Es sollte noch erwähnt werden, dass der Nameserver, der die Anfrage bekommt, nicht selbst entscheiden kann, welche Art von Auflösung er verwendet. Dies wird von dem Anfragenden vorgegeben, indem ein entsprechendes Bit in dem DNS-Paket gesetzt wird, wenn eine rekursive Anfrage gewünscht ist. Da ein Resolver keine iterativen Anfragen bearbeiten kann, setzt er das Bit bei jeder Anfrage, während Nameserver untereinander meist iterativ kommunizieren. Im Folgenden werden die beiden Möglichkeiten beschrieben.

2.4.2 Iterative Auflösung

Die erste Möglichkeit wird iterative Auflösung genannt und funktioniert folgendermaßen: Der Nameserver überprüft, ob ihm die gesuchte IP-Adresse bekannt ist. Ist das nicht der Fall, so sucht er nach Domain-Namen die dem gesuchten Namen ähnlich sind, also dieselbe Endung haben. Dann gibt er die Adressen der Nameserver zurück, die für diese Domain zuständig sind. Damit hat der Nameserver seine Aufgabe erfüllt und der Anfragende muss einen oder mehrere andere Nameserver kontaktieren, um die gesuchte IP-Adresse zu erfahren. Welcher Nameserver von dem Anfragenden als nächstes beansprucht wird, hängt von dessen Roundtrip Time ab. Die Roundtrip Time (RTT) ist ein Maß dafür, wie lange ein entfernter Nameserver braucht, um auf Anfragen zu antworten [Liu02]. Der RTT-Wert wird durch einen Timer gemessen und nach jeder Anfrage an diesen Server aktualisiert. Muss sich nun ein Nameserver zwischen mehreren anderen entscheiden, so wählt er den mit der kürzesten Roundtrip Time.

2.4.3 Rekursive Auflösung

Die zweite Möglichkeit ist eine rekursive Auflösung. In diesem Fall wird von dem Nameserver erwartet, dass er die gesuchte IP-Adresse liefert oder sollte dies nicht möglich sein, z.B. wenn der Domain-Name nicht existiert, dann sollte er eine Fehlermeldung zurückgeben. Deshalb muss der Nameserver versuchen, die gewünschte IP-Adresse zu erfahren. Dazu könnte er entweder ebenfalls eine rekursive Anfrage an einen anderen Nameserver stellen und damit die Verantwortung, die IP-Adresse herauszufinden, abgeben oder er versucht mit mehreren iterativen Anfragen diese selbst herauszubekommen. Üblicherweise wird die iterative Methode verwendet. Im Folgenden wird anhand von Abbildung 3 die geläufigste Form der Namensauflösung beschrieben, bei der sowohl rekursiv als auch iterativ gearbeitet wird. Ein lokaler Nameserver erhält eine rekursive Anfrage von einem Clienten, der die IP-Adresse von www.uni-karlsruhe.de benötigt (Ziffer 1). Der lokale Nameserver kontaktiert darauf hin einen Root-Nameserver und fragt diesen nach der IP-Adresse zu diesem Domain-Namen (Ziffer 2). Der Root-Nameserver verweist dann an den zuständigen Top-Level-Domain-Nameserver, indem er dessen IP-Adresse an den Anfragenden zurück gibt (Ziffer 3). Der nächste Schritt ist also, eine Anfrage an den Nameserver der entsprechenden Top-Level-Domain zu senden. Da die Domain www.uni-karlsruhe.de im deutschen Teil des Namensraumes liegt, wird in diesem Beispiel der TDL-Nameserver für .de kontaktiert (Ziffer 4). Auch dieser wird ihm wiederum die Adresse eines anderen Nameservers geben, der spezifischere Informationen zu der gesuchten Domain besitzt. Nun muss natürlich an diesen Nameserver eine Anfrage gesendet werden. In diesem Beispiel ist es der autoritative Nameserver mit dem Domain-Namen netserv.rz.uni-karlsruhe.de (Ziffer 6). Da diesem die Domain www.uni-karlsruhe.de bekannt ist, kann er die entsprechende IP-Adresse an den lokalen Nameserver zurückgeben (Ziffer 7). Hätte der Nameserver die Domain nicht gekannt, so wäre die iterative Namensauflösung solange fortgesetzt worden, bis der lokale Nameserver an den Nameserver geraten wäre, der die gesuchte Antwort zurückgeben können hätte. Sobald der lokale Nameserver die Antwort erhalten hat, kann er die gesuchte IP-Adresse an den Clienten zurückgeben (Ziffer 8) und die Namensauflösung ist beendet. Dieser Prozess kann durch das Caching verkürzt werden, da dadurch mit den iterativen Anfragen nicht jedes mal bei dem Root-Nameserver begonnen werden muss, sondern direkt bei den spezialisierteren Nameservern mit der Anfrage angefangen werden kann.

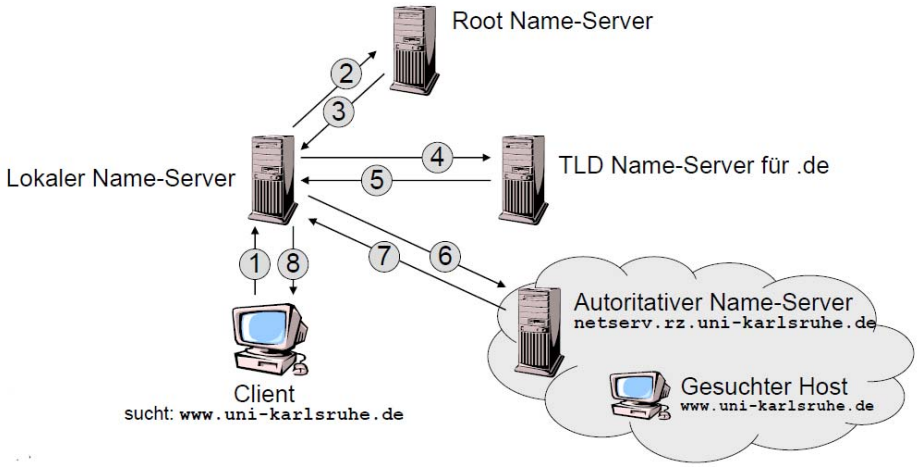

Abbildung 3: Auflösung eines Domain-Namen [Inst08]

2.4.4 Caching und TTL

Um den Prozess der Namensauflösung zu beschleunigen, wurden die Nameserver mit Caches versehen. Wenn ein Nameserver eine rekursive Anfrage bearbeiten muss, erhält er von anderen Nameservern Daten über die gesuchte Domain. Diese Informationen und die IP-Adresse des Domain-Namens können dann in dem Cache gespeichert werden. Auch wenn die gesuchte Domain nicht existiert kann das im Speicher festgehalten werden, dies nennt man negatives Caching. Der große Vorteil des Caching ist, dass gespeicherte Daten schneller abgerufen und sofort zurückgegeben werden können, d.h. dass die Geschwindigkeit bei späteren Anfragen erhöht werden kann.

Wie lange ein Server gesammelte Daten im Cache behält entscheidet der Administrator der für die Zone zuständig ist, in der der Server arbeitet. Die Gültigkeitsdauer der Daten wird in der so genannten TTL (Time to Live) festgelegt. Die TTL gibt die Zeitspanne an, über die der Nameserver die Daten im Cache halten darf. Wenn die TTL überschritten wird, muss der Nameserver die Daten aus dem Cache entfernen und erneut vom verantwortlichen Nameserver laden [Liu02].

3 Schwachstellen und Angriffsmöglichkeiten

In diesem Kapitel sollen die Schwachstellen des Domain Name Systems anhand verschiedener Angriffsmöglichkeiten erläutert werden.

3.1 Cache Poisoning

Angriffe auf den Cache eines Nameservers oder eines Clients, werden allgemein als Cache Poisoning bezeichnet. Da bei so einem Angriff der Cache mit falschen Daten versehen, also „vergiftet“, wird. Bevor Kaminsky Anfang des Jahres die neue Angriffsmöglichkeit entdeckt hat, waren zwei Angriffsmöglichkeiten bekannt.

3.1.1 Vergiften des Anhangs

Die erste bereits bekannte Möglichkeit Einträge in einem Nameserver-Cache zu verändern, beruhte auf der Tatsache, dass Antworten auf eine Anfrage zusätzliche Informationen als Anhang erlaubten. Ein Angreifer brauchte also nur eine Antwort auf eine gestellte Anfrage senden und konnte beliebig Zusatzinformationen, wie Webseiten mit gefälschten IP-Nummern anhängen. Dazu musste jedoch der Server erst eine Anfrage an einen von ihm kontrollierten Nameserver schicken. So eine Anfrage kann erzwungen werden. Eine Möglichkeit wäre es z.B. gewesen eine E-Mail an ein Opfer zu schreiben, die ein Bild enthält, das auf einem Rechner in seinem Domainbereich abgelegt ist. Das gleiche könnte man natürlich auch über eine öffentlich zugängliche Plattform wie z.B. eBay oder ein Forum machen, was in der heutigen Zeit mit Sicherheit einfacher wäre, da im Normalfall E-Mails mit unbekanntem Absender misstraut wird. Diese Verbindung ist nicht ersichtlich. Dadurch wird jedoch eine DNS-Anfrage an den nächsten Nameserver gesendet, um die IP-Adresse für diese Verlinkung herauszubekommen. Durch eine rekursive Anfrage erreicht er irgendwann den Server des Angreifers, der ihm die gesuchte IP-Adresse liefert. Im Anhang der Antwort findet sich ein Domain-Name mit gefälschter IP-Adresse. Dieser wird, wie auch die gesuchte Domain, im Cache des Servers gespeichert. Hat der Domain-Name aus dem Anhang bereits im Cache existiert, wird der alte Eintrag durch die neuen Daten überschrieben und der Cache Poisoning Angriff ist beendet. Zukünftig wird nun bei jeder Anfrage eines anderen Nameservers nach diesem Domain-Namen die falsche

IP-Adresse weitergegeben, bis die TTL des Eintrags abgelaufen ist. Dadurch können viele Internetnutzer auf eine falsche Seite umgeleitet werden und ihnen so eventuell wichtige Daten entwendet werden.

3.1.2 Ausnutzung der ungesicherten Verbindung

Die zweite Möglichkeit eines Cache Poisoning Angriffs ist auf die unsichere Verbindung zurückzuführen, die im Domain Name System benutzt wird. Mit unsicherer Verbindung ist gemeint, dass ein Nameserver nicht überprüfen kann woher die Antwort auf seine Anfrage stammt. Tatsächlich könnte eine Antwort von jedem beliebigen Server kommen. Die einzige Absicherung der Nachrichten ist die Transaktionsnummer (ID). Diese sollte ursprünglich eigentlich nur verwendet werden, um einer Anfrage eine Antwort zuordnen zu können, doch es erschwert auch einem Angreifer das Cache Poisoning. Ohne Transaktionsnummer könnte er jede Anfrage auf Anhieb beantworten und wäre mit Sicherheit wesentlich schneller als die Auflösung der richtigen IP-Adresse durch das DNS. Da ein Nameserver allerdings nur die Antwort mit der richtigen Transaktionsnummer akzeptiert, muss der Angreifer die richtige Transaktionsnummer wissen. Diese IDs sind 16 Bits lang und werden zufällig vergeben, weshalb es 65536 verschiedene Möglichkeiten gibt. Das ist wenig, weshalb die Chancen eine ID zu erraten hoch sind. Dies kann sich der Angreifer zunutze machen, indem er eine Anfrage nach einem bestimmten Domain-Namen an einen Nameserver schickt und diesen darauf hin, sofort mit gefälschten Antworten überhäuft. Da die Namensauflösung Zeit braucht, werden viele seiner Antworten, vor der richtigen Antwort ankommen. Stimmt die ID einer seiner Antworten mit der ID der Anfrage überein, so wird die Antwort akzeptiert und die Daten im Cache gespeichert. Im Normalfall funktioniert diese Methode nicht beim ersten Versuch, so dass die richtige Antwort (also die des befragten Nameservers) in den Cache kommt, wo sie solange gespeichert bleibt wie ihre TTL besteht. Der Angreifer kann den Angriff also erst wiederholen, wenn die TTL abgelaufen ist, da der Nameserver sonst die korrekte Antwort aus dem Cache holen würde. Da es durch diese Methode mehrere Jahre dauern könnte, bis eine ID tatsächlich übereinstimmt, wurde dieses Verfahren nicht als Bedrohung angesehen. Allerdings gibt es verschiedene Ausprägungen dieses Verfahrens, die z.B. Erweiterungen zur Vorhersage besitzen. Mit dem Birthday Paradox (s. Abschnitt 3.2), werden weniger Versuche gebraucht, um eine ausreichende Chance zu erhalten, die ID zu erraten.

3.1.3 Bailiwick Check

Die erste Angriffsmethode, das Vergiften des Anhangs (s. Abschnitt 3.1.1) konnte durch eine Erneuerung der Software vor einpaar Jahren behoben werden. Die neue Software enthielt den so genannten Bailiwick Check (dt: Zuständigkeitsbereichprüfung), mit dem fortan der Anhang einer Antwort überprüft werden konnte. Es werden nur noch Informationen in Anhängen akzeptiert, die in direkter Verbindung mit der angefragten Domain stehen und alle anderen, die Informationen zu fremden Domains enthalten, verworfen. Dadurch wurde diese Methode des Cache Poisoning deutlich erschwert. Die Sicherheit der zweiten Methode, die Ausnutzung der ungesicherten Verbindung (s. Abschnitt 3.1.2), wurde dagegen nicht verbessert, da die Erfolgschancen dieser Methode eher gering waren. Man glaubte, dass diese Angriffsmöglichkeiten wirkungslos seien, bis Kaminsky eine Möglichkeit fand, mit einer Kombination dieser beiden Angriffsmöglichkeiten einen neuen Cache Poisoning Angriff durchzuführen.

3.1.4 Cache Poisoning nach Kaminsky

Bei Kaminskys neuem Verfahren wird eine Anfrage an einen Nameserver gesendet, der einen Domain-Namen enthält, der innerhalb einer bestimmten Domain liegen müsste, aber nicht

existiert. Der Nameserver würde dann, wie bei der vorhergehenden Methode, eine Anfrage stellen um die IP-Adresse herauszubekommen. Da der Domain-Name nicht existent ist, kann er ihn nicht im Cache haben. Während der Server versucht die Adresse zu ermitteln, wird er wiederum mit gefälschten Antworten überhäuft. Wenn die Anzahl der Antworten steigt, die vor der richtigen ankommen, dann steigt auch die Wahrscheinlichkeit, dass eine der IDs mit der ID der Anfrage übereinstimmt und die Antwort akzeptiert wird. Sollten trotzdem alle Antworten des Angreifers fehlschlagen, so ist die erste Antwort, die akzeptiert wird, die, die durch die Namensauflösung ermittelt wurde, und diese wird dann als Eintrag im Cache gespeichert. Damit wäre dieser Domain-Name für den Angreifer nicht mehr zu gebrauchen, da er erst warten müsste bis die TTL abgelaufen ist, bevor er den Angriff wiederholen kann, was sehr zeitintensiv ist. Deshalb wird er stattdessen einen weiteren nicht existenten Domain-Namen aus dieser Domain benutzen und mit diesem den Angriff neu beginnen. Auf diese Weise kann er den Vorgang so oft wiederholen, wie er will. Letztendlich wird irgendwann eine Antwort akzeptiert werden. Kaminsky selbst hat in einem Interview nach der Black Hat Konferenz gesagt, dass es ungefähr 10 Sekunden dauert bis eine der Antworten akzeptiert wird. Das ist aber nur der eine Teil des Angriffes, der andere besteht aus den Zusatzinformationen, die wie bereits erwähnt, an eine Antwort angehängt werden dürfen. Würde man einen beliebigen Domain-Namen anhängen, so würde dieser vom Bailiwick Check verworfen werden. Benutzt man jedoch nur Domain-Namen, die innerhalb der selben Domain liegen, wie der Domain-Name nach dem gesucht wurde, so werden diese akzeptiert und im Cache gespeichert. Bereits bestehende Einträge werden mit den neuen IP-Adressen überschrieben. Damit ist der Cache „vergiftet“ und wird auf eine Anfrage von einem anderen Nameserver nach dem Domain-Namen mit der gefälschten IP-Adresse antworten.

3.2 Birthday Paradox

Das bereits erwähnte Birthday Paradox, beschreibt die Wahrscheinlichkeit in einer Menge von Personen zwei Menschen zu finden, die denselben Geburtstag haben. Da es sehr unwahrscheinlich ist, dass eine wahllos auf der Straße angesprochene Person, den selben Geburtstag (Tag und Monat) wie man selbst hat, nämlich nur ungefähr 0,3 %, schätzen viele Menschen die Wahrscheinlichkeit in einer Gruppe von Personen, zwei Menschen mit dem selben Geburtstag zu finden, falsch ein. Tatsächlich ist es aber so, dass die Wahrscheinlichkeit zwei Menschen mit demselben Geburtstag in einer Gruppe von 23 Personen zu finden, bei 50% liegt. Dies scheint verblüffend zu sein, dennoch basiert dieser Wert auf einer rein mathematischen Wahrscheinlichkeitsberechnung. Würde man eine der Personen nach ihrem Geburtstag fragen, so ist die Wahrscheinlichkeit, dass diese Person am selben Tag wie man selbst Geburtstag hat, wiederum sehr gering nämlich 1/365 (rund 0,3%). Fragt man nun eine weitere Person nach deren Geburtstag, so ist die Wahrscheinlichkeit, dass sie mit einer der anderen Personen am selben Tag Geburtstag hat bei ungefähr 0,8%. Mit steigender Anzahl der befragten Personen, steigt also auch die Chance, dass sich zwei Menschen finden lassen, die am selben Tag Geburtstag haben. So sind es bei 30 Personen bereits rund 70% und bei 50 Personen bereits um die 99%.

Das Birthday Paradox lässt sich in der Informatik vielseitig einsetzen, wie z.B. zum Auffinden von Kollisionen in Hashfunktionen. In dieser Arbeit wird jedoch betrachtet wie mit Hilfe des Birthday Paradox die Chancen eine ID zu erraten erhöht werden können. Im Kapitel Cache Poisoning (s. Abschnitt 3.1) wurde erläutert, dass zur Zuordnung von Anfrage und Antwort eine 16-stellige ID verwendet wird, die man braucht, damit eine Antwort akzeptiert wird. Das Birthday Paradox besagt, dass mit steigender Anzahl der Personen die Chance zwei Menschen mit dem selben Geburtstag zu finden stark ansteigt. Dieses Wissen kann für Angriffe genutzt werden, denn es bedeutet, dass bei steigender Anzahl von DNS-Anfragen und Antworten, auch die Chance steigt, dass die IDs von Anfrage und Antwort übereinstimmen. Angenommen ein

Angreifer würde 100 gefälschte Anfragen an einen Nameserver richten und ihm anschließend 100 gefälschte Antworten schicken, so steigt die Chance einer „Kollision“ von Anfrage und Antwort ID von annähernd 0,0 % auf 7,3 %. Würde man statt 100 Anfragen und Antworten 650 nehmen, so würde man mit einer Wahrscheinlichkeit von 96% eine zur Anfrage passende Antwort bekommen.

3.3 Denial of Service

Unter Denial of Service (DoS) Angriff versteht man einen Angriff auf einen Server mit dem Ziel, dessen Dienste nicht mehr verfügbar zu machen. Bei einem DoS Angriff auf einen Nameserver des Domain Name Systems würde somit die Namensauflösung gestört werden. Werden mehrere Nameserver einer Domain gleichzeitig angegriffen, so kann es passieren, dass diese Domain für mehrere Stunden komplett ausfällt und damit in diesem Zeitraum nicht mehr erreichbar ist. Beliebte Angriffsziele sind Nameserver von Root- und Top-Level-Domains.

Ein erfolgreicher DoS Angriff bewirkt eine gezielte Überlastung des Systems. Diese kann bei einem Nameserver durch eine zu hohe Anzahl an Anfragen entstehen. Da ein normaler Nameserver nicht zwischen einer wirklichen und einer gefälschten Antwort unterscheiden kann, muss er jede Anfrage bearbeiten. Der Angreifer muss also nur eine größere Anzahl an Anfragen an den Nameserver senden, um diesen zu überlasten. Sobald der Nameserver nicht mehr alle Anfragen bearbeiten kann, lässt er einige wahllos fallen. Wenn nun ein anderer Nameserver oder Client versucht, an diesen eine Anfrage zu stellen, kann er diese nicht beantworten, da er noch durch die gefälschten Anfragen beschäftigt ist und verwirft die Anfrage. Es ist keine Namensauflösung über diesen Nameserver mehr möglich und somit war der Angriff erfolgreich.

3.3.1 Distributed Denial of Service

Der verteilte Denial of Service (DDoS) ist eine häufig verwendete Ausprägung des DoS. Diese Angriffe sind wesentlich wirkungsvoller als normale Denial of Service Angriffe, da der Angriff von mehreren Systemen aus zur gleichen Zeit stattfindet. Erst am 21.11.2008 wurde ein DDoS Angriff auf die Nameserver des deutschen Registrars und Hosters InternetX ausgeführt. Dadurch wurde die Namensauflösung in dessen Domain schwer gestört. Viele seiner Nameserver, sowohl primäre als auch sekundäre, fielen aus. Viele Internetseiten waren schwer oder gar nicht erreichbar. Zwei bis drei Stunden nach Beginn des Angriffs betrug das Angriffsvolumen ca. 40.000 Hosts und eine Gesamtbandbreite von 20Gbit/s [Bach08], [Ihle08].

Ein Angriff in diesem Ausmaß ist möglich, indem der Angreifer den Angriff auf verschiedene Systeme im Internet verteilt. Diese Systeme stehen bereits durch ein Botnet unter der Kontrolle des Angreifers. Soll nun ein DoS Angriff gegen ein bestimmtes Ziel geführt werden, kann der Angreifer alle Systeme koordiniert starten. Diese stellen dann, wie bei einem normalen DoS Angriff, eine größere Anzahl an Anfragen an die Server des Opfers, mit dem Ziel diese zu überlasten. Durch die Flut der Anfragen, sind die Nameserver nicht mehr in der Lage andere Anfragen zu beantworten. Die Namensauflösung in dieser Zone ist nicht mehr möglich und die Domains des Opfers sind für einige Zeit nicht mehr erreichbar.

3.3.2 DDoS durch DNS Rekursion

Ein ähnlicher Angriff kann auch über falsch konfigurierte rekursive Nameserver geführt werden. Jeder Nameserver lässt standardmäßig DNS-Rekursion zu. Wird diese Funktion bei der Konfiguration nicht ausgeschaltet, so bietet der Server ein Ziel für Denial of Service Angriffe. Diese Server werden häufig bei DDoS Angriffen als Verstärker eingesetzt. Der Angreifer

übernimmt die Identität des Opfers und schickt mehrere tausende rekursive Anfragen an verschiedene Nameserver. Meistens werden dabei gezielt Anfragen gestellt, deren Antworten wesentlich größer sind, als die gesendeten Anfragen. Durch diese Art von Angriff können „Fluten“ von mehreren Gigabit erreicht werden, da durch die Rekursion der Angriff verstärkt wird. Der Angriff wird wahrscheinlich die gesamte Bandbreite des Opfers belegen, selbst wenn diese um ein Vielfaches größer ist als die Bandbreite des Angreifers, so dass das System des Opfers nicht mehr erreichbar ist. Jedoch gefährdet dieser Angriff nicht nur das System des Opfers, sondern den gesamten Teil des Netzes, in dem der DDos Angriff ausgeführt wird.

Zum Abschluss dieses Kapitels, soll in Abbildung 4 ein Überblick über die Angriffsmöglichkeiten im DNS, anhand des Bildes aus Abschnitt 2.3 gegeben werden. Die erste Möglichkeit im DNS Schaden anzurichten sind nicht autorisierte Updates (roter Pfeil ganz links). Dabei werden von dem Angreifer Updates an den Master-Nameserver gesendet. Dieser verändert seine Daten entsprechend dem Update und arbeitet fortan mit den gefälschten Daten. Der Angreifer könnte jedoch auch die Identität des Master vortäuschen (zweiter roter Pfeil von links). Dadurch könnte er alle zur Zone gehörende Slaves mit falschen Daten versorgen. Da diese ihre Zonendaten vom Master durch Zonentransfer beziehen. Übernimmt er dagegen die Identität des Caching Forwarders (roter Pfeil ganz rechts), so kann er unbemerkt falsche Antworten an einen Resolver schicken. Der Cache Poisoning Angriff (s. Abschnitt 3.1)(roter Pfeil mittig), kann sowohl bei Master als auch Slaves durchgeführt werden.

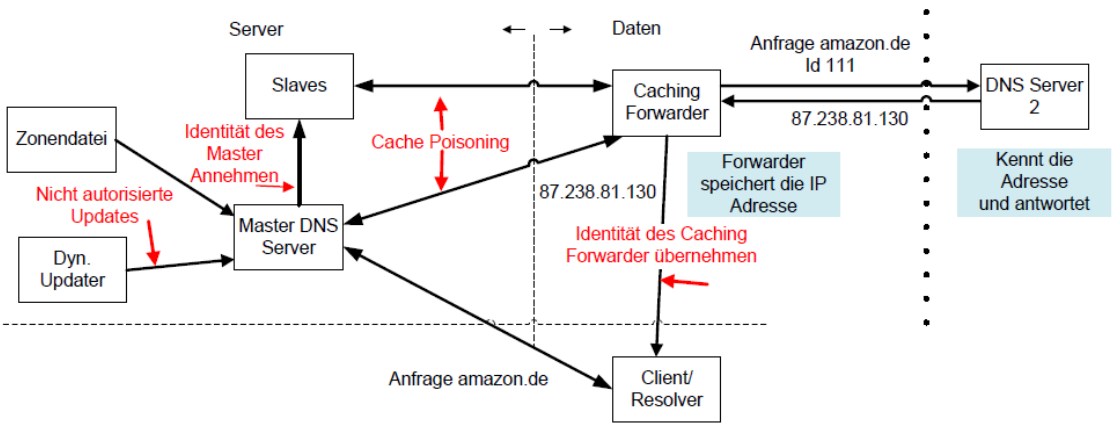

Abbildung 4: Angriffsmöglichkeiten im DNS

4 Schutzmaßnahmen

Das Ausmaß und die Folgen dieser Angriffe verdeutlichen wie wichtig mehr Sicherheit im Domain Name System wäre. Wird eine neue Schwachstelle im DNS entdeckt, so wird versucht diese durch Patches zu schließen. Da jedoch für deren Entwicklung oftmals nicht viel Zeit bleibt, sind diese meist keine Garantie dafür, dass die Schwachstelle nicht mehr ausgenützt werden kann, auch wenn sie dies wesentlich erschweren. Eine langfristige Lösung wäre hier deshalb sinnvoller. Diese könnte durch eine Erweiterung der jetzigen DNS Software erreicht werden, in diesem Kapitel wird dazu ein möglicher Ansatz gezeigt.

4.1 Patches

Patches sollen Schwachstellen oder Fehler in Softwaresystemen beheben. Ihre Entwicklung beginnt sobald eine Schwachstelle oder ein Fehler entdeckt wurde, um schnell Abhilfe zu

schaffen. Doch oftmals kann das Problem durch Patches nicht behoben sondern nur verkleinert werden. Ein Beispiel dazu ist Kaminsky's Patch, der entwickelt wurde um Systeme, vor der von ihm entdeckten Cache Poisoning Methode (s. Abschnitt 3.1.4) zu schützen. Dieser Patch beruht darauf, dass nicht nur die ID-Nummern zufällig verteilt werden, sondern ebenso die benutzten Portnummern. Da die Portnummer ebenfalls 16-Bit lang ist, müsste der Angreifer nun 32 Bits erraten. Da ist möglich, allerdings braucht es dafür mehrere Milliarden Anfragen. Dadurch würde jede Menge Datenverkehr erzeugt, die einem aufmerksamen Netzwerkadministrator nicht entgehen würde. Es ist also dank des Patches wesentlich schwerer, aber nicht unmöglich diesen Angriff auszuführen.

4.2 DNSSEC

DNSSEC ist eine Erweiterung von DNS, deren Ziel Datenauthentizität und Datenintegrität ist. Sie wird in den RFCs 4033 [AALM+05a], 4034 [AALM+05c] und 4035 [AALM+05b] definiert. Durch DNSSEC kann eine empfangene Nachricht authentifiziert werden und dadurch kann das Cache Poisoning weitgehend unterbunden werden. Für die Authentifikation wird die Public-Key-Verschlüsselung verwendet.

4.2.1 Signieren und Verifizieren einer Nachricht

Da die Public-Key-Verschlüsselung nicht nur relativ sicher ist, sondern auch das Problem, der Schlüsselverteilung und Verwaltung von symmetrischen Verschlüsselungsverfahren bei großen Systemen, löst, wird sie in der Sicherheitserweiterung für das DNS eingesetzt. Jedoch würde es viel zu lange dauern jedes Mal die komplette Nachricht zu verschlüsseln, da asymmetrische Verschlüsselungsalgorithmen in der Regel ziemlich langsam sind. Deshalb wird stattdessen mit Hilfe einer Hashfunktion ein Hashwert über die Nachricht gebildet und dieser, mit dem privaten Schlüssel, asymmetrisch verschlüsselt. Dieser Vorgang ist in Abbildung 5 (oben) graphisch dargestellt. Das Ergebnis davon wird digitale Signatur genannt und an die Nachricht angehängt. Anschließend wird die Nachricht versendet. In Abbildung 5 (unten) wird erläutert, wie die Identität des Absenders überprüft werden kann. Dazu muss die Signatur mit dem öffentlichen Schlüssel des Senders entschlüsselt werden, um den Hashwert zu bekommen. Anschließend muss er, von der erhaltenen Nachricht, den Hash berechnen. Dazu muss dieselbe Hashfunktion wie die des Absenders verwendet werden. Deshalb wird das verwendete Hashverfahren mit der Nachricht übertragen. Stimmen nun der erhaltene Hashwert und der selbst berechnete Hashwert miteinander überein, so stammt die Nachricht von dem Absender. Durch dieses Verfahren kann die Nachricht zwar nicht vor unbefugter Einsichtnahme geschützt werden, jedoch vor Änderung durch einen Dritten.

4.2.2 Spezielle Resource Records

Im RFC 4034 [AALM+05c] wurden vier spezielle Resource Records (RRs) für DNSSEC definiert: RRSIG, DNSKEY, DS und NSEC. In RRSIG Resource Records werden die digitalen Unterschriften (Signaturen) gespeichert. Dadurch können die RRSIG Resource Records die zur Zone gehörenden Resource Records signieren. Das hat den Vorteil, dass dann bei einer Antwort auf eine Anfrage überprüft werden kann, ob die RRs verändert wurden und ob sie von der erwarteten Zone stammen. Dieser Authentifizierungsprozess ist allerdings nur möglich, wenn ein öffentlicher Schlüssel existiert, mit dem es möglich ist die Signatur der RRs zu verifizieren. Dieser öffentliche Schlüssel wird von einem weiteren Resource Record, DNSKEY genannt, gespeichert und verwaltet. Ein Resolver kann dann die Informationen des DNSKEY

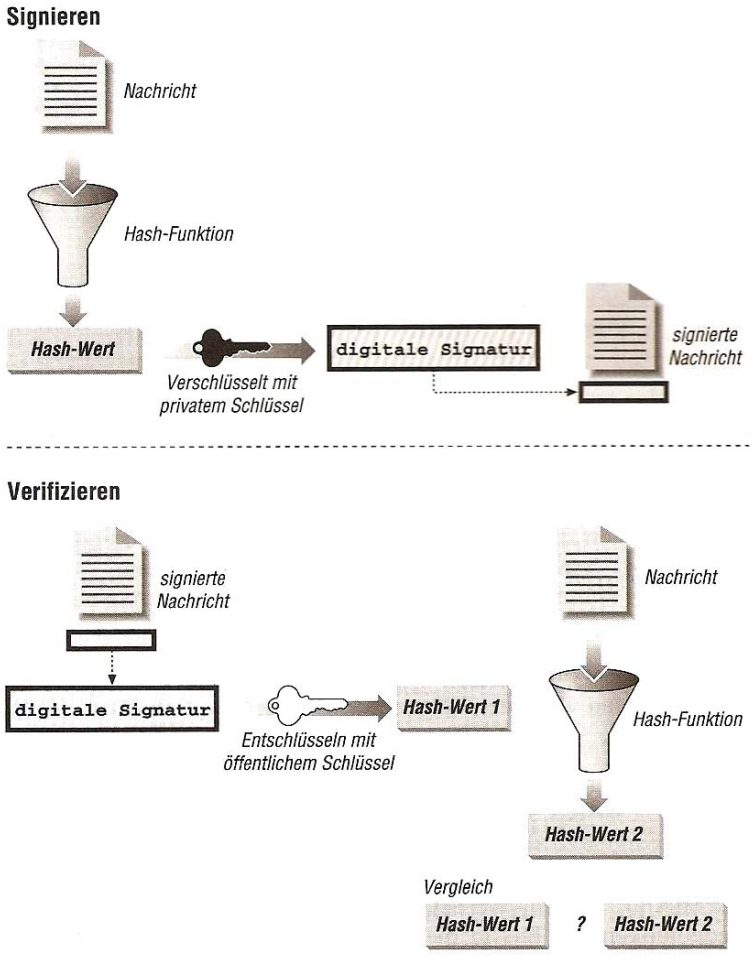

Abbildung 5: Signieren und Verifizieren einer Nachricht [Liu02]

benutzen um die Signatur einer Zone zu verifizieren. Der DS Resource Record ist dafür zuständig verschiedene DNSSEC Zonen miteinander zu verbinden. Dazu verweist der DS auf den DNSKEY einer untergeordneten Zone, indem er die Daten des DNSKEY speichert. Da diese Zone wiederum DS RRs enthalten kann, die auf weitere Zonen verweisen, entsteht eine Chain of Trust (s. Abschnitt 4.2.3).

Der vierte RR wird NSEC genannt. Mit diesem RR kann man beweisen, dass ein bestimmter Domain Name innerhalb einer Zone nicht existiert. Dazu werden alle Labels einer Zone alphabetisch sortiert. Anschließend wird für jedes Label der Liste ein NSEC Resource Record erzeugt. Dieser beinhaltet Informationen über die für diesen Resource Record definierten Typen und dessen Nachfolger. Als Nachfolger des letzten Eintrags wird der erste Eintrag angegeben. Dadurch werden alle existierenden Resource Records einer Zone miteinander verkettet. Anschließend wird jeder NSEC durch den aktuellen RRSIG RR signiert, damit sie nicht unbemerkt verändert werden können. Bei einer Anfrage nach einem bestimmten Domain-Namen wird der entsprechende Eintrag zurückgegeben. Existiert der Domain-Name jedoch nicht, so wird der Eintrag zurückgegeben, der die beiden Labels verkettet, zwischen denen der gesuchte liegen müsste. Somit kann eindeutig verifiziert werden, dass der gesuchte Eintrag nicht existiert. Allerdings hat dieser Resource Record auch einen großen Nachteil. Durch sukzessive Anfragen lassen sich in kürzester Zeit alle Informationen über eine Zone auslesen. Dazu braucht ein Angreifer nur solange Anfragen nach nicht existierenden Domain-Namen senden, bis er alle Resource Records der verketteten Liste ausgelesen hat. Dieser Vorgang

wird Zone Walking oder auch DNSSEC Walking genannt. Dies erleichtert dem Angreifer die Bestimmung von Angriffszielen. Um dieses Problem zu umgehen wurde Anfang 2008 im RFC 5155 [LSAB08] ein neuer Resource Record NSEC3 eingeführt, der als Alternative zu NSEC eingesetzt werden kann. Dieser kann ebenfalls eindeutig verifizieren, dass ein Eintrag nicht existiert. Das Zone Walking ist jedoch bei diesem Resource Record nicht mehr möglich.

4.2.3 Chain of Trust

Die Chain of Trust ist ein Modell, dass die Vertrauensbeziehungen zwischen Domains verschiedener Ebenen definiert. Dieses besteht, wie der Domain-Namensraum, aus einer Baumstruktur mit Zonen. Um zu verstehen, wie die Chain of Trust aufgebaut ist, muss man etwas über die Schlüssel, die in DNSSEC verwendet werden, wissen. Es gibt zwei Arten von Schlüsseln, die jeweils als privater und öffentlicher Schlüssel existieren, den Key Signing Key (KSK) und den Zone Signing Key (ZSK). Ein Administrator der seine Zone sichern will, gibt seinen KSK an den Administrator der nächst höheren Domain und lässt ihn von ihm signieren. Mit dem signierten Schlüssel kann er dann seine ZSKs signieren, mit denen sich wiederum die Daten der Zone signieren lassen. Dieser Vorgang setzt sich im Baum nach unten fort, dadurch entsteht eine Kette des Vertrauens (Chain of Trust). Die Menge aller Zonen, die durch einen einzigen Schlüssel gesichert sind, wird als Sicherheitsinsel (Island of Security) bezeichnet [Aitc05].

4.2.4 Nachteile von DNSSEC

Allerdings hat DNSSEC auch Nachteile. Die Umstellung kostet viel Geld, da die Systeme entsprechend aktualisiert werden müssen. Derzeitig genutzte Resolver können mit den aufwendigen kryptographischen Berechnungen nicht umgehen und müssen deshalb ausgetauscht werden. Alternativ kann man auch nur einen neuen Resolver benutzen über den alle alten Modelle kommunizieren, was jedoch ein Sicherheitsrisiko in sich birgt. Bei den Nameservern reicht meistens ein Update auf DNSSEC. Jedoch kann es sein, dass auch diese Hardware, wegen den erhöhten Anforderungen erneuert werden muss. Auch viele DSL- und WLAN-Router haben Schwierigkeiten mit DNSSEC umzugehen. Ein weiterer Nachteil ist der höhere Verwaltungs- und Wartungsaufwand der für Administratoren anfällt, da die Schlüssel regelmäßig erneuert werden müssen, um die Sicherheit zu gewährleisten. Ein anderer negativer Aspekt ist, dass durch DNSSEC DoS Angriffe nicht verhindert werden können, sondern sogar durch den höheren Aufwand für kryptographische Berechnungen und die wesentlich größeren Zonendateien vereinfacht werden.

5 Zusammenfassung

Abschließend kann festgehalten werden, dass das Domain Name System eine kritische Infrastruktur darstellt, bei der einzelne Teile durch gezielte Angriffe leicht ausgeschaltet oder manipuliert werden können. Da es das Internet, in seiner bekannten Form, ohne das DNS nicht geben würde, ist es wichtig, dieses vor Manipulation zu schützen. DNSSEC geht bereits in die richtige Richtung. Jedoch hat es noch zu viele Schwachstellen, um einen sicheren Schutz bieten zu können. Sobald jedoch die entsprechenden Randbedingungen geschaffen werden, hat DNSSEC das Potential das Internet ein Stück sicherer zu machen.

Literatur

[AALM+05a] R. Arends, R. Austein, M. Larson, D. Massey und S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), März 2005.

[AALM+05b] R. Arends, R. Austein, M. Larson, D. Massey und S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), März 2005. Updated by RFC 4470.

[AALM+05c] R. Arends, R. Austein, M. Larson, D. Massey und S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), März 2005. Updated by RFC 4470.

[Aitc05] Ron Aitchison. *Pro DNS und BIND*, Kapitel 11. Springer-Verlag New York. 2005.

[ArMi07] Suranjith Ariyapperuma und Chris J. Mitchell. Security vulnerabilities in DNS and DNSSEC. In *Availability, Reliability and Security*, Vienna, April 2007. S. 335–342.

[Bach08] Daniel Bachfeld. DDoS-Attacke auf InternetX [Update]. Webseite, November 2008. `http://www.heise.de/newsticker/meldung/print/119274`.

[EaKa97] D. Eastlake 3rd und C. Kaufman. Domain Name System Security Extensions. RFC 2065 (Proposed Standard), Januar 1997. Obsoleted by RFC 2535.

[East97] D. Eastlake 3rd. Secure Domain Name System Dynamic Update. RFC 2137 (Proposed Standard), April 1997. Obsoleted by RFC 3007.

[Fang06] Fanglu Guo and Jiawu Chen and Tzi-cker Chiueh. Spoof Detection for Preventing DoS Attacks against DNS Servers. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*. IEEE Computer Society, 2006, S. 37–37.

[Ihle08] Jens Ihlenfeld. DDoS Angriff auf DNS Provider. Webseite, November 2008. `http://www.golem.de/0811/63704.html`.

[Inst08] Institut für Telematik. Anwendungssyteme. Kommunikation und Datenhaltung Vorlesung, 2008.

[Kami08] Dan Kaminsky. Catching up with Kaminsky. *Network Security* 2008(9), September 2008, S. 4–7.

[KöKu07] Kösel und Kurgzell. *DNS/DHCP*, Kapitel 6 and 7. Open Source Press. 2007.

[Liu02] Paul Albitz & Cricket Liu. *DNS und BIND*, Kapitel 1. O'Reilly. 2002.

[LSAB08] B. Laurie, G. Sisson, R. Arends und D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155 (Proposed Standard), März 2008.

[Mark04] Adam Greenhalgh Mark Handley. Steps towards a DoS-resistant internet architecture. In *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future Directions in Network Architecture*, New York, NY, USA, August 2004. ACM, S. 49–56.

[Math08] Frank H. Mathis. A Generalized Birthday Problem. *SIAM Review (Society for Industrial and Applied Mathematics)* 33(2), Juli 2008, S. 265– 270.

[Mock83a] P.V. Mockapetris. Domain names: Concepts and facilities. RFC 882, November 1983. Obsoleted by RFCs 1034, 1035, updated by RFC 973.

[Mock83b] P.V. Mockapetris. Domain names: Implementation Specification. RFC 883, November 1983. Obsoleted by RFCs 1034, 1035, updated by RFC 973.

[Mock87a] P.V. Mockapetris. Domain names – concepts and facilities. RFC 1034 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592.

[Mock87b] P.V. Mockapetris. Domain names – Implementation and Specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343.

[Schn08] David Schneider. Fresh phish. *IEEE Spectrum* 45(10), Oktober 2008, S. 34–38.

[US-C06] US-CERT (Hrsg.). The Continuing Denial of Service Thrat Posed by DNS Recursion *v*2.0. White Paper, US–Cert, 2006.

[VTRB97] P. Vixie, S. Thomson, Y. Rekhter und J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), April 1997. Updated by RFCs 3007, 4035, 4033, 4034.

[WeIh06] S. Weiler und J. Ihren. Minimally Covering NSEC Records and DNSSEC On-line Signing. RFC 4470 (Proposed Standard), April 2006.

[Wiss08] Ulrich Wisser. ...Kontrolle ist besser. *c't* Band 14, 2008, S. 202– 207. Soft-Link 0814202.

[YKMC06] Lihua Yuan, K. Kant, P. Mohapatra und Chen-Nee Chuah. DoX: A Peer-to-Peer Antidote for DNS Cache Poisoning Attacks. In *Communications, 2006. ICC '06. IEEE International Conference on*, Band 5, Istanbul, Juni 2006. S. 2345–2350.

Abbildungsverzeichnis

Identity Management und Single-Sign-On

Martin Merkel

Diese Arbeit behandelt die grundlegenden allgemeinen Aspekte des Identity Managements, das dabei hilft digitale Identitäten zu verwalten, eine zentrale Benutzer- und Rechteverwaltung aufzubauen und die Komplexität bestehender Systeme zu verringern. Möglichkeiten zur technischen Realisierung eines solchen Dienstes werden ebenso vorgestellt wie das Konzept des Single-Sign-Ons. Als populäres Beispiel für SSO wird abschließend noch das Kerberos-Protokoll vorgestellt.

1 Einleitung

Sicherheit in der IT spielt in der heutigen Zeit eine immer wichtigere Rolle. Gerade bei großen Unternehmen herrscht ein reges Kommen und Gehen bei den Mitarbeitern. Für die Administratoren bedeutet das oftmals einen hohen Verwaltungsaufwand. Benutzerzugänge müssen erstellt und gelöscht werden, die Berechtigungen gesetzt bzw. widerrufen werden und die Datenbestände auf allen Systemen synchron gehalten werden. Eine zentrale Datenbank für diesen Zweck existiert meistens nicht, was oftmals einfach daran scheitert, dass die Dienste eine solche nicht unterstützen. Die Administratoren haben also die schwere Aufgabe dafür zu sorgen, dass alle Mitarbeiter ihre tägliche Arbeit verrichten können und gleichzeitig niemand unberechtigt die Unternehmenssysteme benutzen kann. Aber auch auf Mitarbeiterseite könnte die Situation einfacher sein. Sie müssen sich ständig neu an den Systemen anmelden, sich mehrere Benutzernamen und Passwörter merken und diese zuordnen können.

Identity Management versucht diese Probleme durch ein zentrales System zu vereinfachen. Sind alle Daten - Benutzerkonten und ihre zugehörigen Berechtigungen - dort gespeichert, können Dienste ihre Informationen von dort beziehen und ihre jeweils interne Datenbank mit Benutzerinformationen auflösen. Administratoren müssen sich also nicht mehr um die Konsistenz der Daten zwischen mehreren Systemen kümmern und Mitarbeiter können mit ihrem Benutzernamen und einem Passwort alle Systeme im Rahmen ihrer Befugnisse benutzen.

Was ein Identity-Management-System aber nicht alleine lösen kann, ist das Problem der ständigen Anmeldungen. Der Mitarbeiter hat zwar nur noch ein Passwort, muss es aber trotzdem noch mehrmals eingeben. Hier kommt das Single-Sign-On-Konzept ins Spiel. Ein Benutzer sich dabei nur einmal an einem zentralen System anmelden und kann ab dann ohne weitere Passworteingabe alle anderen Dienste benutzen.

Im ersten Teil dieser Arbeit werden nun die grundlegenden und einige technische Aspekte des Identity Managements vorgestellt. Themen wie wirtschaftlicher Nutzen oder gesetzliche Forderungen werden nicht behandelt. Diese werden zum Beispiel ausgiebig in [Mezl08] besprochen. Anschließend wird im zweiten Teil das Single-Sign-On-Konzept vorgestellt.

2 Identity Management

In diesem Abschnitt werden nun grundlegende Themen des Identity Managements behandelt und Ansätze für eine technische Realisierung vorgestellt.

2.1 Ziel und Anforderungen

Ein Identity-Management-System muss mehrere Anforderungen erfüllen. Dazu zählen u.a.

- Zentrale Speicherung der Daten

  Sämtliche für das Identity Management relevante Daten können in einem zentralen System gespeichert werden.

- Verwaltung von Benutzerkonten und deren Passwörtern

  Benutzerkonten werden nur noch zentral im Identity-Management-System erstellt, bearbeitet und gelöscht.

- Verwaltung von Zugriffsberechtigungen

  Alle Zugriffsberechtigungen werden im Identity-Management-System verwaltet.

- Schnittstelle zur Überprüfung Benutzerdaten für Dienste

  Sämtliche Dienste führen keine eigene Datenbank mehr, sondern prüfen Zugangsdaten gegen die im Identity-Management-System hinterlegten Daten.

Durch diese Forderungen bleibt die Konsistenz der Daten gewahrt, die Übersichtlichkeit steigt und die Komplexität sinkt im Vergleich zu Systemen, bei denen die Daten an vielen verschiedenen Orten gespeichert werden und in Folge dessen konsistent gehalten werden müssen.

Das Hauptziel eines Identity-Management-Systems ist Sicherheit. Die Mitarbeiter sollen dabei ihre Tätigkeiten durchführen und Unbefugte keinen Zugriff auf Systeme und Daten bekommen können. Weitere Ziele wären zum Beispiel eine Kostenreduktion zu erreichen oder gesetzlichen Anforderungen nachzukommen. Siehe dazu auch [DiHa08] und [Mezl08].

2.2 Komponenten des Identity Managements

Damit ein Benutzer einen Dienst benutzen kann, muss er sich am Dienst anmelden. Diese Anmeldung besteht allerdings immer aus mehreren Schritten. Der Dienst überprüft dabei zunächst die Identität des Benutzers auf Basis der Daten, die er bei der Anmeldung bekommen hat. Anschließend stellt er sicher, dass der Benutzer den Dienst auch wirklich in Anspruch nehmen darf.

Es sind also drei Dinge für eine Anmeldung notwendig:

- Eine *digitale Identität* des Benutzers,
- die Überprüfung der Identität oder *Authentifizierung* und
- die Überprüfung der Benutzerrechte oder *Autorisierung*

Diese grundlegenden Komponenten werden nun in den kommenden Abschnitten vorgestellt.

2.3 Digitale Identität

Identität ist, was etwas eindeutig von allen anderen Dingen unterscheidbar macht. Das heißt, dass eine Identität eine Menge von Merkmalen ist, die ausreicht, um etwas eindeutig zu bestimmen. Dabei kann “etwas” sowohl auf Menschen als auch auf alles andere, wie z.B. Computer, bezogen werden.

Digital bedeutet, dass die Merkmale der Identität digital erfassbar sein müssen. So kommen beispielsweise die Merkmale *Name*, *Geburtsdatum*, *Geburtsort* und *Adresse* für die Identität einer Person in Frage.

2.4 Authentifizierung

Mit jeder digitalen Identität ist ein Bezeichner, z.B. der Benutzername, verbunden. Anhand dieses Bezeichners kann ein Benutzer eine Identität und die damit verbundenen Rechte für sich beanspruchen.

Authentifizierung ist nun der Teil der Anmeldung, in dem geprüft wird, ob ein Benutzer auch der ist, für den er sich ausgibt.

In der deutschen Sprache gibt es außerdem noch den Begriff der Authentisierung. Dieser bedeutet das Nachweisen der eigenen Identität. Im Englischen wird diese Unterscheidung nicht vorgenommen, hier heißt es in beiden Fällen "authentication".

Der Benutzer übermittelt also zuerst den Bezeichner der Identität an den Dienst, den er benutzen möchte. Der Dienst fragt dann je nach Sicherheitsstufe ein oder mehrere Merkmale der Identität ab. Dabei unterscheidet man die drei folgenden Arten [DiHa08]:

- Wissensbasiert

  Der Benutzer identifiziert sich hierbei durch etwas, was nur er weiß. Das Passwort ist dabei die häufigste Art der Authentifizierung.

- Besitzbasiert

  Hier muss der Benutzer etwas besitzen. Beispielsweise eine Smartcard mit einem darauf gespeicherten Zertifikat.

- Eigenschaftsbasiert oder biometrisch

  Der Benutzer wird anhand von seinen körperlichen Eigenschaften identifiziert, wie z.B. seinem Fingerabdruck

2.5 Autorisierung und Zugriffskontrolle

Autorisierung beschreibt den Vorgang, durch den ein Benutzer nach Überprüfung seiner Zugriffsrechte in die Lage versetzt wird, auf einen Dienst zuzugreifen.

Davon zu unterscheiden ist die Zugriffskontrolle. Sie legt fest, wer welche Befugnisse hat. Dabei wurden im Laufe der Zeit mehrere Ansätze entwickelt. Zu nennen sind dabei [DiHa08]:

- Discretionary access control (DAC)

  Bei *DAC* werden die Berechtigungen *durch die Benutzer* festgelegt. Das Modell basiert auf dem Gedanken, dass ein Eigentümer die Rechte seiner Objekte selber festlegen darf. Die Linux/UNIX-Dateiberechtigungen sind ein Beispiel für *DAC*, denn der Eigentümer einer Datei kann den Zugriff auf diese über die Felder "owner", "group" und "other" selber festlegen.

- Mandatory access control (MAC)

  Bei *MAC* werden die Berechtigungen über *globale Richtlinien vom Administrator des Systems* festgelegt. Durch Attribute werden Benutzer und andere Objekte (Dateien, Dienste, etc.) in verschiedene Klassen eingeteilt. Die Zugriffsmöglichkeiten auf jeweils andere Klassen sind dann durch die globalen Richtlinien entsprechend eingeschränkt.

- Chinese wall policy oder Brewer-Nash-Modell

  Die *Chinese wall policy* versucht, Interessenskonflikte zu unterbinden. Wenn ein Benutzer neue Berechtigungen erhält, wird deswegen geprüft, welche Tätigkeiten er bis dahin ausgeführt hat und ob die Informationen, die er aus der alten Tätigkeit besitzt, für seine

neuen Aufgaben einen Interessenskonflikt erzeugen. Im diesem Falle würde der Benutzer die neuen Berechtigungen nicht erhalten können. Die *Chinese wall policy* findet vor allem im Banken- und Finanzwesen Anwendung, um z.B. Insiderhandel zu vermeiden. Siehe auch [BrNa89] und [Ecke07].

- Role-based access control (RBAC)

  Das *rollenbasierte RBAC* ist das heute am häufigsten eingesetzte Modell. Der Grund dafür ist, dass es in einem Unternehmen meist wesentlich mehr Mitarbeiter als Rollen gibt und der Verwaltungsaufwand somit geringer ist. Eine Rolle fasst Berechtigungen zusammen, die ein Benutzer braucht, um eine bestimmte Tätigkeit auszuführen. Die Rolle "Mitarbeiter" könnte einen Benutzer beispielsweise in die Lage versetzen, sich an verschiedenen Arbeitsstationen anmelden und Emails lesen zu können. Eine weitere Rolle "Administrator" würde dem Benutzer erlauben, Änderungen an den Systemen durchzuführen. Rollen können auch hierarchisch aufgebaut werden. So sind Unterscheidungen nach Abteilung, etc. möglich.

  Für *RBAC* gibt es mittlerweile viele Erweiterungen wie *ABAC* (Attribute based access control), *RB-RBAC* (Rule based access control) oder *PBAC* (Policy based access control), die es erlauben Rollen dynamisch anhand von Attributen zuzuweisen. Gerade für große Unternehmen mit einer Vielzahl von Beschäftigten sind diese interessant. Siehe dazu [DiHa08].

2.6 Technische Realisierung

Um ein Identity-Management-System aufzubauen, muss es eine Möglichkeit geben, sämtliche Informationen zu den Benutzern und Diensten zentral zu speichern. Zu diesem Zweck wurden Verzeichnisdienste geschaffen. Sie speichern ihre Daten im Gegensatz zu anderen Datenbankarten in einer hierarchischen Struktur (basierend auf dem X.500 Standard der ITU-T[1]) und sind besonders für lesenden Zugriff optimiert, da die Informationen wesentlich häufiger abgerufen als verändert werden. Für den Zugriff wird in der Regel ein Directory Access Protocol (DAP) oder die leichtgewichtigere Variante LDAP (Leightweight DAP) verwendet. Es erlaubt Einträge zu erstellen, zu ändern und zu löschen. Außerdem stellt es Such- und Vergleichsmethoden zur Verfügung. Abbildung 1 stellt den grundlegenden Aufbau eines solchen Verzeichnisses schematisch dar. Die Abkürzung "dc" steht in einer LDAP-Verzeichnishierarchie für *domain component*, "ou" für *organizational unit*. Das Beispielverzeichnis enthält die Daten der *FirmaXY* (Wurzel "dc=FirmaXY,dc=de") und ist in drei Grundbereiche für Mitarbeiter, Inventar und Dienste aufgeteilt. Die Mitarbeiter werden nochmals nach Abteilungen getrennt im Verzeichnis geführt. Um die Daten eines Mitarbeiters namens "Meier" abzufragen, muss ein Lesebefehl für den Eintrag "cn=Meier,ou=Verkauf,ou=Mitarbeiter,dc=FirmaXY,dc=de" abgeschickt werden ("cn" steht für common name).

Die Dienste haben durch das Verzeichnis die Möglichkeit ihre eigene Datenbank aufzulösen und alle Informationen, die sie zur Authentifizierung der Benutzer benötigen, vom zentralen Speicher zu beziehen. Problematisch bleibt dabei nur die Methode, wie sich der Benutzer gegenüber dem Dienst identifizieren kann. Um ständige Anmeldungen zu vermeiden und Passwörter durch das Netzwerk zu schicken, wird in Verbindung mit Identity Management oft versucht, Single-Sign-On zu realisieren. Dieses Konzept wird im nächsten Abschnitt vorgestellt.

[1] Internationale Fernmeldeunion

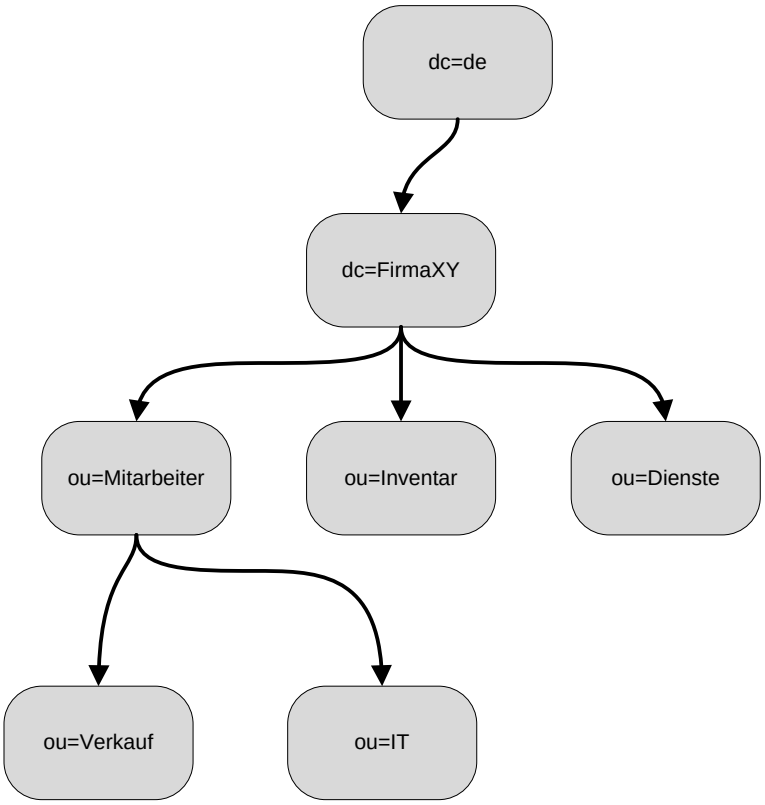

Abbildung 1: Aufbau eines LDAP-Verzeichnisses.

3 Single-Sign-On

Die Idee bei Single-Sign-On (SSO) ist, dass sich ein Benutzer nur noch ein einziges Mal an einer zentralen Stelle anmelden muss und ab dann alle Systeme im Rahmen seiner Befugnisse benutzen kann. Die Dienste brauchen sich also auch nicht mehr um eine Authentifizierung kümmern, denn der Single-Sign-On-Mechanismus garantiert die Identität des Benutzers. Dies bringt einige Vorteile mit sich, wie z.B.

- Die Benutzer müssen sich nur noch ein Passwort merken
- Die Benutzer können gezwungen werden ein qualitativ hochwertiges Passwort zu verwenden anstatt mehrerer schwächerer
- Das Ändern der Attribute und Berechtigungen sowie das Sperren eines Benutzers ist einfach zu handhaben

Auf der anderen Seite muss aber auch dafür gesorgt werden, dass das SSO-System immer verfügbar ist, da sonst kein Anmeldevorgang mehr funktioniert.

Das prominenteste Beispiel für ein Single-Sign-On-System ist das Kerberos-Protokoll, das im folgenden Teil näher beschrieben wird.

3.1 Das Kerberos-Protokoll

Der Ursprung des Kerberos-Protokolls ist das Massachusetts Institute of Technology (MIT). Dort veröffentlichten Steve Miller und Clifford Neumann in den späten 1980ern Version 4 des Protokolls, das hauptsächlich für ihr Projekt “Athena” vorgesehen war. Version 1-3 waren bis

dahin nur MIT interne Versionen. 1993 wurde Version 5 von John Kohl und Clifford Neuman als RFC veröffentlicht [KoNe93]. 1996 veröffentliche das MIT Version 1.0 ihrer Kerberos v5 Implementierung. Ein Jahr später gab Microsoft bekannt, Kerberos in seiner nächsten Windows-Version zu benutzen. Seit dem sind viele Ergänzungen am Protokoll vorgenommen worden, u.a. erschien 2005 RFC 4120 "The Kerberos Network Authentication Service (V5)", das das ursprüngliche RFC 1510 als "obsolete" deklariert [NYHR05].

Kerberos stellt einen Authentifizierungsdienst bereit und spezifiziert ein Verfahren auf Basis des Needham-Schroeder-Protokolls [NeSc78] zur sicheren Authentifizierung von Benutzern über ein Netzwerk. Kerberos benutzt dazu symmetrische Verschlüsselung und braucht eine vertrauenswürdige dritte Partei. Diese wird durch das KDC - das *Key Distribution Center* - zur Verfügung gestellt. Das KDC besteht wiederum aus dem *Authentication Server* (AS) und dem *Ticket Granting Server* (TGS). Kerberos basiert auf einem Ticketsystem, das zum Beweis der Identität eines Benutzers dient. Alle Kerberos-Teilnehmer - Benutzer sowie Server - werden als "principal" bezeichnet. Jedem *principal* wird ein Schlüssel zugewiesen, der in der Schlüsseldatenbank des KDC gespeichert ist. Der Ablauf einer Sitzung wird im folgenden Abschnitt anhand eines Beispiels beschrieben.

3.2 Ablauf einer Kerberossitzung

Das folgende Beispiel beschreibt wie sich Alice per Kerberos an einen Mailserver anmelden will. Zum leichteren Verständnis wird dabei das Kerberos V4 Protokoll verwendet.

Grob skizziert sieht der Ablauf folgendermaßen aus: Um irgendeinen Dienst benutzen können, wird zuerst immer ein *Ticket Granting Ticket* (TGT) vom *Authentication Server* benötigt. Dieses Ticket stellt den Identitätsnachweis für alle weiteren Aktivitäten dar. Anschließend muss beim *Ticket Granting Server* ein Ticket für den jeweiligen Dienst, den man benutzen möchte, beantragt werden. Hat man dieses erhalten, so schickt man es an den Dienst, der den Zugriff daraufhin erlaubt.

Im Detail sind die folgenden sechs Schritte, die auch in Abbildung 2 dargestellt sind, notwendig bis Alice den Mailserver benutzen kann [DiHa08]:

1. **Beantragung eines TGT**

   Alice gibt ihre Benutzerdaten (Name und Passwort) in ein lokales Kerberos-Clientprogramm ein. Dieses generiert daraufhin den MD5-Hash des Passworts, der Alices Master Key K_A darstellt. Das Programm sendet nun eine Nachricht an den *Authentication Server*, die ausschließlich aus dem Wort "Alice" im Klartext besteht. Das Passwort sowie der MD5-Hash verlassen ihren Rechner nicht.

2. **Antwort des AS mit dem TGT**

   Der *Authentication Server* prüft nach Erhalt der Nachricht, ob Alice in der Benutzerdatenbank existiert. Wenn ja, generiert er einen Session Key S_A. Da Alices Passwort im Klartext in der Benutzerdatenbank gespeichert ist, kennt auch der *AS* Alices Master Key K_A. Er sendet nun eine mit K_A verschlüsselte Nachricht zurück an Alice. Diese besteht inhaltlich aus zwei Teilen:

   - Dem Session Key S_A und
   - dem *Ticket Granting Ticket* (TGT), das wiederum das Wort "Alice" und den Session Key S_A enthält. Das TGT ist dabei für Alice allerdings nicht lesbar, denn es ist mit dem Schlüssel K_{TGS} des *Ticket Granting Server* (TGS) verschlüsselt.

   Die ersten beiden Schritte sind nur ein einziges Mal notwendig. Das *TGT* ist die Grundlage für Alice beim *TGS* Tickets für Dienste beantragen zu können.

3. **Beantragung des Service Tickets**

   Das Clientprogramm entschlüsselt nun mit Alices Master Key die Antwort des *AS* und hat somit den Session Key S_A und das mit K_{TGS} verschlüsselte TGT in seinem Besitz. Da Alice jedoch den Mailserver benutzen will, sendet das Clientprogramm jetzt eine Nachricht an den *TGS*, die aus drei Teilen besteht:

   - Dem Wort "mailserver",
   - dem verschlüsselten TGT und
   - dem sog. *Authenticator*, der seinerseits aus dem Wort "Alice" und einem Zeitstempel besteht. Der *Authenticator* wird dabei mit dem Session Key S_A verschlüsselt und dient der Authentisierung gegenüber dem *TGS*. Der Nachweis der Identität wird dadurch erbracht, dass nur Alice den *Authenticator* erzeugt haben kann, da nur sie im Besitz des Session Keys ist.

4. **Antwort des TGS mit dem Service Ticket**

   Der *TGS* entschlüsselt die Nachricht von Alice und hat somit den Authenticator und das immer noch verschlüsselte TGT. Letzteres wird nun ebenfalls dekodiert, wodurch der *TGS* den Besitz des Session Keys S_A erlangt. Jetzt kann auch der Authenticator lesbar gemacht werden, wodurch der *TGS* den Zeitstempel überprüfen kann, um etwaige Replay-Attacken auszuschließen. Der *TGS* erzeugt nun einen Session Key S_{AM}, den Alice später für die Kommunikation mit dem Mailserver verwenden muss. Sie bekommt wieder ein mit S_A verschlüsseltes Paket aus zwei Teilen:

   - Dem Session Key S_{AM} verschlüsselt mit S_A und
   - dem Service-Ticket verschlüsselt mit dem Schlüssel des Mailservers K_M. Es besteht aus dem Wort "Alice" und dem erzeugten Session Key S_{AM}.

5. **Anmeldung beim Dienst**

   Das Clientprogramm empfängt die Antwort des *TGS* und entschlüsselt sie wieder. Es hat nun das Service-Ticket und nach weiterem Decodieren auch den Session Key S_{AM}. Damit Alice den Mailserver endlich benutzen kann, sendet das Programm eine letzte, zweiteilige Nachricht an den Mailserver. Sie enthält:

   - Das verschlüsselte Service-Ticket und
   - einen neuen, mit S_{AM} verschlüsselten *Authenticator*, der aus dem Wort "Alice" und einem neuen Zeitstempel besteht.

6. **Antwort des Dienstes**

   Der Mailserver entschlüsselt nun das Service-Ticket und ist somit im Besitz des Session Keys S_{AM}. Hiermit kann er den Authenticator lesbar machen, den Zeitstempel prüfen und wieder eine Attacke unterbinden. Damit Alice noch sichergehen kann, dass sie mit dem echten Mailserver kommuniziert, schickt dieser ein abschließendes, mit S_{AM} verschlüsseltes Paket, das den um eins imkrementierten Zeitstempel enthält, an sie.

3.3 Trust Domains

Wenn man ein Kerberos Key Distribution Center aufsetzt, muss man dessen Zuständigkeitsbereich, den sog. "Realm" oder die "Domain", festlegen. Principals existieren also immer nur innerhalb einer Domain. Durch Outsourcing und andere Kooperationen zwischen Unternehmen ist es jedoch oftmals notwendig, dass auch andere Personen, die sich außerhalb der eigenen Domain identifiziert haben, die eigenen Dienste in Anspruch nehmen können.

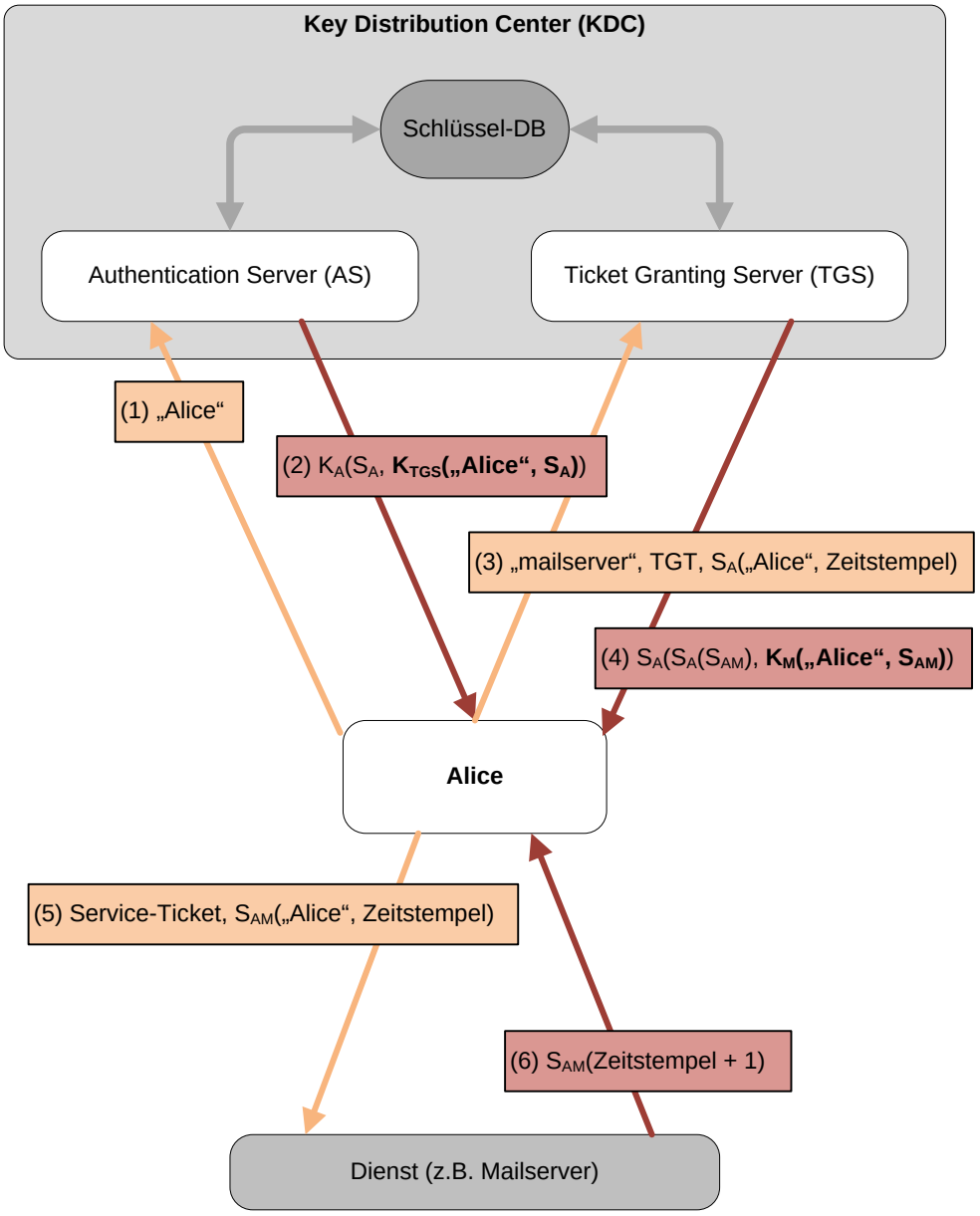

Abbildung 2: Ablauf einer Kerberossitzung.

Für genau diesen Einsatzzweck enthalten Single-Sign-On-Lösungen Mechanismen um anderen Realms zu vertrauen. D.h. ein Benutzer, der zu Realm A gehört und sich dort authentisiert hat, kann die Dienste aus Realm B benutzen, wenn A und B eine Vertrauensbeziehung besitzen. Die Art der Vertrauensstellung kann auf zwei Weisen gegeben sein (siehe auch Abbildung 3).

- Direkte Vertrauensbeziehung:

  Realm A und B kennen und vertrauen sich gegenseitig.
- Transitive oder implizite Vertrauensbeziehung:

  Realm A und B kennen und vertrauen sich gegenseitig. Da B aber noch Realm C vertraut, existiert auch eine implizite Vertrauensbeziehung zwischen A und C.

Es liegt am Unternehmen zu entscheiden, ob man impliziten Vertrauensbeziehungen wirklich vertraut.

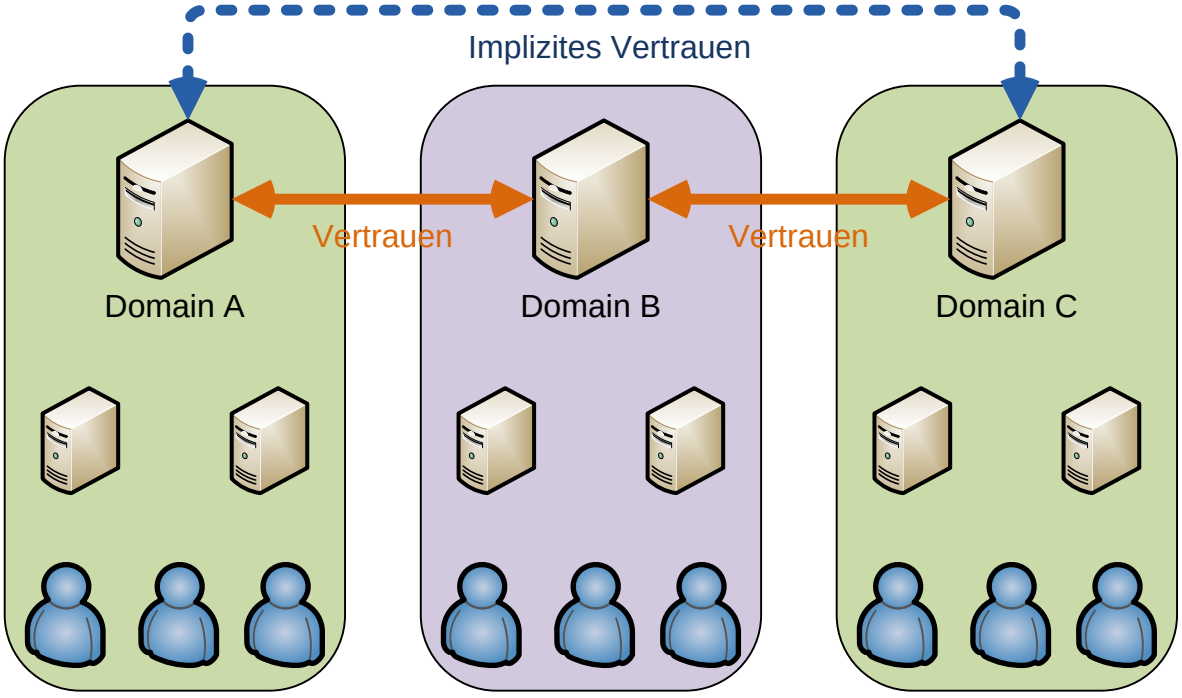

Abbildung 3: Direkte und implizite Vertrauensstellung.

Dienste werden innerhalb einer Domain nach zwei Kriterien differenziert. Wenn ein Dienst für den Nachweis von Identitäten zuständig ist, wird er als Identity-Provider (IdP) bezeichnet. Alle anderen Dienste fasst man unter Service-Providern zusammen. Der oder die IdPs sind also die Repräsentanten einer Domain.

Single-Sign-On in Verbindung mit der Benutzung von herkömmlichen Diensten wie Email oder FTP ist zwischen Trust Domains mit Kerberos kein Problem mehr. Da jedoch zunehmend versucht wird Kosten einzusparen, werden viele Applikationen nur noch webbasiert angeboten. Auf den Komfort von Single-Sign-On will man aber auch hier nicht verzichten. Innerhalb eines Intranets ist die Lösung mittels Cookies auch echt einfach. Doch auch hier möchte man externen Personen Zugriff ermöglichen. Um die Kommunikation per HTTP zwischen zwei Identity-Providern bzw. einem IdP und einem Service-Provider zu standardisieren, wurde deshalb die "Security Assertion Markup Language" (SAML) entwickelt (Siehe dazu [FuMJ07] und [Gros03]).

Sie spezifiziert ein webbasiertes Single-Sign-On-Verfahren und den webbasierten Austausch von Authentifizierungs- und Autorisierungsdaten zwischen Identity-Providern unterschiedlicher Domains. Der Ablauf wird durch HTTP-Redirects gesteuert und die Informationen im XML[2]-Format in den HTTP-Anfragen mittels SOAP[3] gekapselt.

4 Zusammenfassung und Ausblick

Die Wachsende Zahl an Kooperationen zwischen Unternehmen und der Trend zum Outsourcing machen ein unternehmensweites Identity Management immer wichtiger. Das einfache Erzeugen von Benutzerzugängen und die zentrale Verwaltung der rollenbasierten Rechte sind nur einige der Vorteile. Jedoch müssen Standards geschaffen werden, damit die Funktionstüchtigkeit zwischen den Lösungen verschiedener Hersteller gewährleistet ist. Dies ist zur Zeit

[2]Extensible Markup Language
[3]Simple Object Access Protocol

noch nicht der Fall. Es existieren zahllose proprietäre Lösungen, die jedoch nur mit anderen Produkten desselben Herstellers kompatibel sind. Im Bereich des Single-Sign-Ons sind mit Kerberos und dem noch recht jungen SAML-Standard für Web-SSO zwei Technologien verfügbar, die viel Potenzial besitzen und nur noch eingesetzt werden müssen.

Literatur

[Area04] The Open Group Identity Management Work Area (Hrsg.). Identity Management. White Paper, The Open Group, März 2004.

[BrNa89] D.F.C. Brewer und M.J. Nash. The Chinese Wall security policy. *1989 IEEE Symposium on Security and Privacy. Proceedings.*, 1989, S. 206–214.

[DiHa08] Jochen Dinger und Hannes Hartenstein. *Netzwerk- und IT-Sicherheitsmanagement*. Universitätsverlag Karlsruhe. 2008.

[Ecke07] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg. 2007.

[FuMJ07] S. Fugkeaw, P. Manpanpanich und S. Juntapremjitt. Adding SAML to two-factor authentication and single sign-on model for dynamic access control. *6th International Conference on Information, Communications and Signal Processing*, 2007, S. 1–5.

[Gros03] T. Gross. Security analysis of the SAML single sign-on browser/artifact profile. *19th Annual Computer Security Applications Conference*, 2003, S. 298–307.

[KoNe93] J. Kohl und C. Neuman. The Kerberos Network Authentication Service (V5). RFC 1510 (Proposed Standard), September 1993. Obsoleted by RFC 4120.

[Mezl08] Christian Mezler-Andelberg. *Identity Management*. dpunkt.verlag. 2008.

[NeSc78] R.M. Needham und M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM, 21(12)*, 1978, S. 993–999.

[NYHR05] C. Neuman, T. Yu, S. Hartman und K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), Juli 2005. Updated by RFCs 4537, 5021.

Abbildungsverzeichnis