Kryptographische Anonymisierung bei Verkehrsflussanalysen

Andreas Grinschgl
a.grinschgl@student.tugraz.at

Institut für Angewandte Informationsverarbeitung
und Kommunikationstechnologie (IAIK)
Technische Universität Graz
Inffeldgasse 16a
8010 Graz, Österreich

Diplomarbeit

Betreuer: Dipl.-Ing. Dr.techn. Mario Lamberger
Dipl. Ing. (FH) Johannes Weinzerl / C.C.COM

Oktober, 2010

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am 13.10.2010

Andreas Grinschgl

Danksagung

Zu allererst möchte ich meinen Betreuern Mario Lamberger und Johannes Weinzerl für ihre Unterstützung bei dieser Diplomarbeit danken. Besonderer Dank ergeht an Ursula und an meine Eltern, die während der Erstellung dieser Arbeit des Öfteren auf mich verzichten mussten und mir bei der der Korrektur geholfen haben. Weiters möchte ich mich für die Rechtschreibkorrektur dieser Arbeit bei Sonja bedanken. Abschließend möchte ich noch ein Dankeschön an alle meine Freunde, die mir mit Rat und Tat zur Seite standen, richten.

Abstract

Modern traffic flow analysis systems are using some kind of personal data as base for measurements. This could be the license plate or in the case of this master thesis, a blutetooth address. This master thesis is about making this personal data as anonymous as possible without loosing the ability to provide a traffic flow model with this data. This has to be realized by crytopgraphic means.

Keywords: Anonymization, Pseudonyms, Bluetooth, PKI

Kurzfassung

Moderne Systeme zur Verkehrsstromanalyse basieren auf der Erhebung persönlicher Daten. Das könnte ein Nummernschild, oder wie im Falle dieser Diplomarbeit, eine Bluetooth Adresse sein. Diese Arbeit beschäftigt sich nun mit der Aufgabe, die erhobenen Daten soweit wie möglich zu anonymisieren, ohne dabei die Möglichkeit einer Verkehrstromanalyse zu verlieren. Dies soll mit Hilfe kryptographischer Mittel erfolgen.

Stichwörter: Anonymisierung, Pseudonyme, Bluetooth, PKI

Inhaltsverzeichnis

Kapitel 1

Einführung

Bei jeder Art von Datenspeicherung besteht die Frage nach dem Datenschutz. Datenschutz bezieht sich auf die Art der Daten, die gespeichert werden, sowie auf die mögliche Weiterverarbeitung dieser Daten. Oft ist die gesetzliche Lage nicht klar und daher wird versucht ein Maximum an Datenschutz zu erreichen. Zusätzlich ist Datenschutz ein Verkaufsargument für neu entwickelte Systeme. Die Firma C.C.COM steht bei ihrem Projekt BLIDS (siehe Abschnitt 1.2) vor einer solchen Herausforderung. Das Projekt BLIDS beschäftigt sich mit der Verkehrsflusserfassung (Abschnitt 1.3) per Bluetooth. Für den Bereich der Verkehrsflusserfassung sind die gesetzlichen Regelungen in Sachen Datenschutz vage und nur teilweise festgelegt. Das Projekt BLIDS wirft noch weitere Fragestellungen auf: Gehört eine MAC-Adresse zu persönlichen Daten? Wie kann Mißbrauch einer MAC-Adresse in einem Verkehrsflusserfassungssystem verhindert werden? Welche Schlüsse kann man aus einer Verkehrsverfolgung ziehen?

Diese Diplomarbeit beschäftigt sich damit, die MAC-Adresse durch einen anonymen Identifikator zu ersetzen. Die sogenannte Anonymisierung wird dabei aber durch einige Parameter des Systems erschwert. Die verteilte Architektur und die Notwendigkeit eines eindeutigen Identifikators stellen die gravierendsten Schwierigkeiten bei einer Umsetzung dar. Der in dieser Diplomarbeit gewählte Weg stützt sich auf die Verwendung kryptographischer Komponenten wie Public Key Kryptographie und Hash Funktionen um eine Anonymisierung umzusetzen. In Abschnitt 2.7 werden andere Systeme die eine ähnliche Funktion erfüllen betrachtet. (Als Beispiele sind hier E-Voting und E-Money zu nennen.) Weiters werden alle notwendigen kryptographischen Grundkomponenten, die für eine solche Anonymisierung nötig sind, vorgestellt (Kapitel 2). Mit Hilfe dieses Grundwissens werden verschiedene Konzepte, mit denen die Aufgabenstellung gelöst werden kann, entwickelt. Die Vor- und Nachteile werden für jedes Konzept untersucht und so wird versucht eine optimale Lösung zu finden. Aus den gewonnen Daten wird für jede Umsetzung eine Bedrohungsanalyse und eine Liste möglicher Angriffspunkte erstellt. Anschließend wird eine grobe Abschätzung der benötigten Leistung durchgeführt. Das über diese Parameter gewählte System wird dann im Detail ausgearbeitet und für eine Umsetzung im BLIDS System vorbereitet. Dazu wurden die wichtigen Komponenten in Java und C++ implementiert und geeignete Bibliotheken und Komponenten für eine spätere Integration in BLIDS ausgewählt.

1.1 Verkehrserfassung

Die moderne Gesellschaft kennt keinen Stillstand. Ob es nun die fortschreitende technische Entwicklung, den Wandel im Leben von Einzelpersonen oder den Verkehr auf unseren Straßen betrifft, bei allem wird versucht den stetigen Fluss aufrecht zu erhalten. Besonders beim Verkehr ist dies nur durch gezielte Steuerung möglich. Eine solche Steuerung kann nur durch Wissen über die Verkehrssituation und eventuelle Brennpunkte im Verkehrsgeschehen erfolgen. Die Gewinnung der dazu nötigen Daten erfolgt durch verschiedene Erfassungssysteme (vgl. [24]).

1.1.1 Systeme zur Verkehrserfassung

Das am weitesten verbreitete System zur Erfassung der Verkehrssituation ist mit Sicherheit die Induktionsschleife. Dabei wird in den Asphalt der Straße eine Kabelschleife eingelassen. Überfährt oder parkt nun ein Fahrzeug auf dieser Schleife ändert sich die Eigenfrequenz eines mit der Schleife verbundenen Schwingkreises. Dies kann detektiert werden. Solche Induktionsschleifen finden in erster Linie bei der Schaltung von Ampeln, als Auslösung von Rotlichtblitzern oder zur Messung von Fahrzeugfrequenzen Anwendung. Etwas komplexer ist die Fahrzeugerfassung mit Infrarot- oder Radarsystemen. Diese werden entweder an Überkopfwegweisern, Brücken oder Straßenmasten befestigt. Viele dieser Systeme sind so konzipiert, dass sie autonom eingesetzt werden können. Die Stromversorgung erfolgt über Solarpanele und die Daten werden über das GSM Netz übertragen. Bei all den bisher vorgestellten Systemen werden Fahrzeuge nicht identifiziert. Zur Erfassung und Identifizierung von Fahrzeugen haben sich Video-Systeme durchgesetzt (vgl. [24]). Diese werden elektronisch ausgewertet und können Fahrzeuge anhand ihres Kennzeichens identifizieren. Solche Systeme werden oft von der Polizei bei der Fahndung eingesetzt. Um die Verkehrssituation zu erfassen kann auch auf sogenannte kooperative Fahrzeuge zurückgegriffen werden. Diese Fahrzeuge werden mit einer Mess- und Sendeeinheit ausgestattet und übertragen Daten wie Standort, Beschleunigung und Geschwindigkeit an die Verkehrszentrale. Aus den Daten dieser Fahrzeuge kann dann statistisch auf die gesamte Verkehrsituation geschlossen werden. Oft kommen hier Taxis oder Fahrzeuge im öffentlichen Dienst zum Einsatz (vgl. [1]). Zusätzlich existieren luft- und satellitengestütze Systeme (vgl. [23]). Als Alternative zu den bisher vorgestellten Systemen hat die Firma C.C.COM ein System zur Erfassung der Verkehrssituation auf Basis von Bluetooth-Endgeräten entwickelt. Mehr dazu in Abschnitt 1.2.

1.1.2 Verkehrsmodelle

Eine reine Ansammlung von Daten kann noch nicht zur sinnvollen Steuerung von Verkehrsflüssen verwendet werden. Aus den Daten muss ein Modell der aktuellen Situation erstellt werden. Diese Situation kann dann durch gezielte Einflußnahme aufgrund des Modells in eine gewünschte Situation übergeführt werden. Zur Modellierung von Verkehrsdaten existieren zwei verschiedene Ansätze. Auf der einen Seite existieren Systeme zur Modellierung makroskopischer Verkehrsdaten. Dabei werden die Daten einer Gruppe von Fahrzeugen ausgewertet. Mikroskopische Systeme dagegen beruhen auf der Erfassung von Einzelfahrzeugen (vgl. [30]).

1.1.2.1 Makroskopische Verkehrsmodelle

Makroskopische Modelle betrachten nur eine Menge von anonymen Fahrzeugen. Aus diesem Grund kann bei solchen Modellen auch nicht auf die genaue Wegstrecke eines Fahrzeuges rückgeschlossen werden. Die Modellierung verlässt sich in diesem Fall auf Querschnittsmessungen an verschiedenen Stellen eines Verkehrswegs. Der Verkehr wird dabei als Fluss betrachtet, Straßen als Kanäle mit unterschiedlichen Durchflussmengen und Kreuzungen stellen Gabelungen mit unterschiedlich großen Abflüssen dar. Mit dieser Art der Modellierung sind Erkenntnisse über Stauungen, Auslastungen und Durchschnittsgeschwindigkeiten möglich (vgl. [30]).

1.1.2.2 Mikroskopische Verkehrsmodelle

Hier wird der Verkehrsfluss aus der Menge der Einzelfahrzeuge modelliert. Meist ist eine Erfassung aller Fahrzeuge nicht vorgesehen oder nicht möglich. Die Hochrechnung erfolgt daher wieder statistisch. Problematisch bei solchen Systemen ist der Bereich Datenschutz. Sind einzelne Fahrzeuge identifizierbar wird schnell die Angst einer potentiellen Langzeitverfolgung (siehe Abschnitt 3.1) geäußert. Daher nimmt der Umgang mit dem Datenschutz einen immer größeren Stellenwert ein. Aus einer mikroskopischen Modellierung können Daten über Wegstrecken, Auslastungen, Geschwindigkeiten oder Reisezeiten abgeleitet werden (vgl. [30]).

1.1.3 Verkehrserfassung und Datenschutz

Kommen Systeme zum Einsatz, bei denen Einzelfahrzeuge erfasst und identifiziert werden, wirft dies in vielen Fällen Konfliktpunkte im Bereich Datenschutz auf. So ist in vielen Ländern das Speichern von Kennzeichen und Verkehrsdaten nicht erlaubt. In einigen deutschen Bundesländern ist selbst das Aufnehmen und Abgleichen von Kennzeichen durch die Polizei strittig[1]. Ebenso wird der Bereich Datenschutz bei Verkehrserfassungssystemen in der österreichischen Straßenverkehrsordnung STVO §98 [9] erwähnt:

> "Verkehrsbeobachtung
> § 98f. (1) Soweit dies
> 1. für die Regelung sowie die Leichtigkeit, Flüssigkeit und Sicherheit des Verkehrs oder
> 2. für die Erfüllung der den Behörden und Straßenerhaltern gesetzlich obliegenden Aufgaben
> erforderlich ist, dürfen die Behörden und Straßenerhalter zur Beobachtung des Verkehrsgeschehens technische Einrichtungen zur Bildübertragung einsetzen.
>
> (2) Eine bildgebende Erfassung, die eine Identifizierung von Personen oder Fahrzeugen ermöglicht, ist jedoch nur zulässig, soweit dies im Einzelfall zwingend erforderlich ist, um die Aufgaben nach Abs. 1 zu erfüllen.
>
> (3) Eine Speicherung von gemäß Abs. 1 gewonnenen Daten ist nicht zulässig. Für Zwecke der Information der Öffentlichkeit im Wege von Medien dürfen im Bedarfsfall auf Anfrage manuell einzelne Bildquellen ausgewählt und daraus

[1] http://www.focus.de/politik/deutschland/datenschutz_aid_139688.html

kurze Bildfolgen gespeichert und an Medien übermittelt werden, soweit eine Identifizierung von Personen oder Fahrzeugen nicht möglich ist."

In der STVO wird jedoch nur die bildgebende Erfassung erwähnt. Weitere Erfassungsmethoden bei denen ein Fahrzeug identifiziert werden kann, bleiben ausgeklammert. Daher muss, um Konflikte mit dem Gesetzgeber zu verhindern, bei all diesen Systemen besonderer Wert auf die Anonymisierung der erhobenen Daten gelegt werden. Eine komplette Anonymisierung verhindert jedoch eine Verwendung der Daten zur mikroskopischen Verkehrsflussmodellierung, da aus den vorhandenen Daten keine Fahrtstrecken und somit kein Verkehrsfluss ermittelt werden kann. Diese Diplomarbeit beschäftigt sich mit einer Lösung, die dem Datenschutz entspricht und zugleich eine mikroskopische Verkehrsflussmodellierung erlaubt.

1.2 BLIDS

Die Firma C.C.COM hat sich auf Projekte im Bereich der Verkehrstelematik spezialisiert. Eines dieser Projekte ist BLIDS [52]. BLIDS deckt im Individualverkehr die Bereiche Verkehrszählung und Verkehrsflusserfassung ab. Bereits umgesetzte Systeme zur Verkehrszählung basieren meist auf Detektorschleifen. Bei der Verkehrsflusserfassung kommen meist optische Erfassungssysteme zum Einsatz. All diese Systeme sind teuer und wartungsaufwändig. BLIDS wurde als Alternative zu all diesen Systemen entwickelt und basiert auf der Detektion von Bluetooth Endgeräten im Inneren der Fahrzeuge. Diese werden durch Sensorstationen erfasst und die gesammelten Daten anschliessend weiterverabeitet (vgl. [52]).

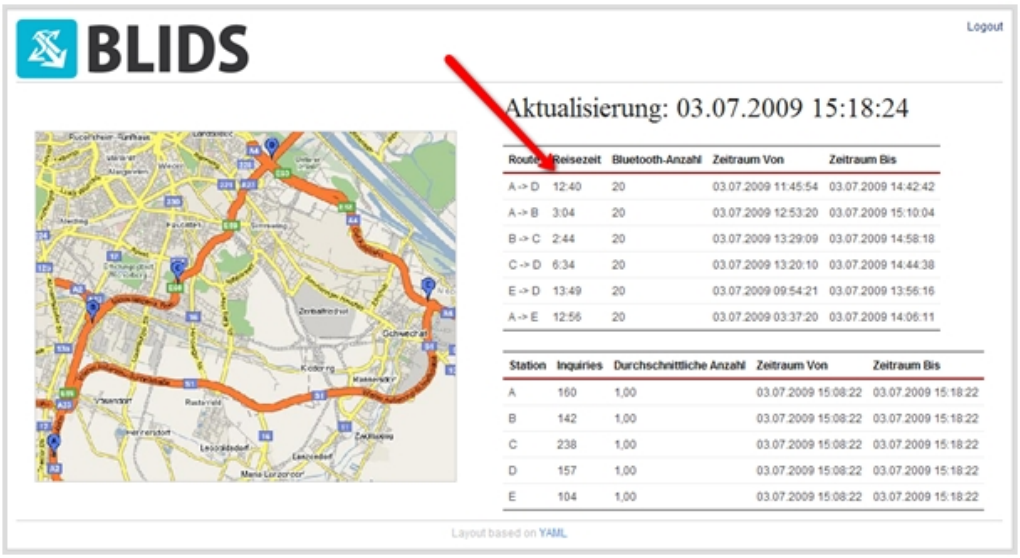

Abbildung 1.1: Darstellung der Reisezeiten durch BLIDS [52]

1.2.1 Datenerfassung

Die Datenerfassung bei BLIDS erfolgt mit Sensorstationen. Über den Verkehrswegen positioniert, erfassen diese Sensorstationen aktivierte Bluetoothgeräte im Inneren der Fahr-

zeuge. Dazu wird von der Sensorstation ein Inquiry (siehe Kapitel Bluetooth 1.3) gestartet. Alle im Sende- und Empfangsbereich befindlichen Endgeräte, die sich zusätzlich im Inquiry-Scan Modus befinden, antworten der Sensorstation mit einem FHS-Paket. Dieses Paket enthält unter anderem die weltweit eindeutige Bluetooth-Adresse des Endgeräts. Die Firma C.C.COM hat die Aufenthaltsdauer eines Endgerätes im Sende- und Empfangsbereich ermittelt, damit eine Erfassung mit hoher Wahrscheinlichkeit noch möglich ist. Es dauert maximal 10,34 Sekunden bis ein Endgerät im Sendebereich der Sensorstation erfasst werden kann. Minimal kann eine Erfassung bereits nach 1,25 ms erfolgen. Der Durchschnittswert für die Erfassung eines beliebigen Endgerätes beträgt 3 Sekunden. Aus diesen Zeitwerten und der vermutlichen Geschwindigkeit der Fahrzeuge wird die Sende- und Empfangsleistung der Sensorstation und damit die Größe des Sende- und Empfangsbereichs abgeleitet. Bei einem Erfassungszeitraum von 10,24 s und einer Fahrzeuggeschwindigkeit von 20m/s ergibt dies eine notwendige Größe von 205m. Zur Erhöhung der Erfassungswahrscheinlichkeit wird dieser Bereich doppelt so groß gewählt. Dies wird durch Anpassung von Sende- und Empfangsleistung sowie durch die Wahl einer geeigneten Antenne erreicht. Andere Faktoren, die den Sende- und Empfangsbereich der Sensorstationen beeinflussen, sind unter anderem bauliche Gegebenheiten, Bäume, hügeliges Gelände oder große Glasfassaden. Um eine korrekte Erfassung sicherzustellen, dürfen sich die Sende- und Empfangsbereiche der Sensorstationen nicht überschneiden (vgl. [52]).

Abbildung 1.2: Typischer BLIDS-Messstellenaufbau [52]

1.2.2 Datenübermittlung

Für die Datenübermittlung stehen verschiedene Methoden zur Verfügung. Die Wahl der favorisierten Methode hängt von der Einsatzart des BLIDS-Systems ab. Soll nur eine Verkehrszählung durchgeführt werden, können die Daten zu einem späteren Zeitpunkt gesammelt an den Server übertragen werden. Ist jedoch eine zeitgerechte Erhebung von Verkehrsflüssen gewünscht, ist es absolut notwendig, die Daten nach der Erhebung sofort an den Server weiterzuleiten. Zu diesem Zweck verfügen die BLIDS-Sensorstationen über ein integriertes EDGE/GPRS Modem. Für die Benutzung dieser Übertragungsmöglichkeit, ist für jede Sensorstation eine SIM-Karte notwendig. Der Datenverkehr wird dabei über eine TCP/IP-Kanal an den Server übermittelt. Die Kosten sind von Mobilfunkanbieter zu Mobilfunkanbieter verschieden, hängen aber meist von der übertragenen Datenmenge ab. Bei einem Verkehrsaufkommen von 150000 Fahrzeugen, kann mit 3 Megabyte an anfallenden Daten gerechnet werden. Als Alternative kann die integrierte Ethernet Schnittstelle genutzt werden. Dies setzt jedoch eine vorhandene Verkabelung voraus. Als Erweiterung wird zusätzlich der Einbau eines WLAN-Moduls angeboten. Die Datenübertragen wird dann über ein bestehendes WLAN realisiert (vgl. [52]) .

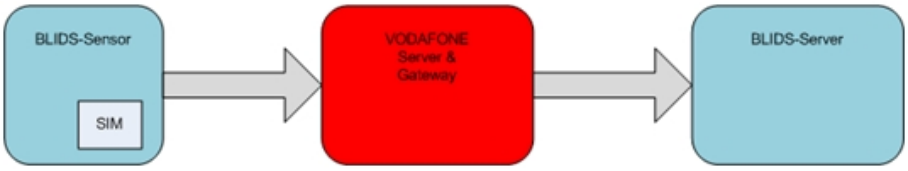

Abbildung 1.3: Datenübermittlung zum BLIDS-Server [52]

1.2.3 Weiterverarbeitung der Daten

Nach der Übertragung der Daten werden diese auf einem Datenbankserver weiterverarbeitet. Die Art der Verarbeitung ist abhängig von der vom Kunden gewünschten Aufgabe des Systems. Wenn nur eine Verkehrszählung durchgeführt werden soll, werden die Daten anonymisiert und in eine SQL-Datenbank eingetragen. Die Anonymisierung in diesem Fall ist einfach durch die Löschung aller personenrelevanten Inhalte zu realisieren. Soll eine Verkehrsflussanalyse durchgeführt werden, müssen die einzelnen Datensätze zueinander korreliert werden können. Eine komplette Anonymisierung ist hier nicht möglich. Die Entwicklung eines passenden Systems ist Aufgabe dieser Diplomarbeit. Nach der Eintragung der Daten in die Datenbank können diese mit statistischen Mitteln gefiltert und von falschen Messwerten befreit werden. Als Abschluss erfolgt eine kundenspezifische Präsentation der Daten (vgl. [52]) .

1.3 Bluetooth

Die IEEE 802.15.1 Schnittstelle (vgl. [45]) (im Weiteren als Bluetooth bezeichnet) wurde bereits 1999 als Ersatz für Kabelverbindungen zu Endgeräten im Nahbereich konzipiert. Der Standard ist seit 2007 in der Version 2.1 verfügbar und in zahlreichen Geräten, wie Notebooks, PDAs und Mobiltelefonen, implementiert. Bluetooth bietet sowohl die Möglichkeit der Datenübertragung, als auch der Sprachübertragung mit QoS (vgl. [43]).

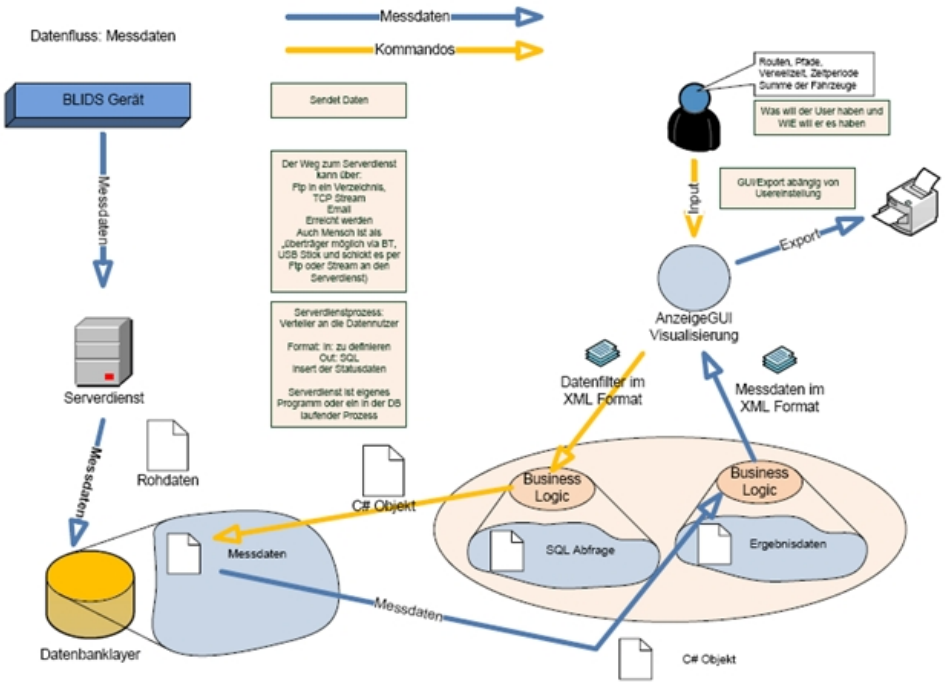

Abbildung 1.4: Weiterverarbeitung der Daten am BLIDS-Server [52]

Bluetooth arbeitet im 2,4 GHz ISM-Band und teilt sich dieses mit diversen anderen Funkschnittstellen. Die vorgesehene Bandbreite beträgt 1 MHz. Bei der Version 1.2 stehen mit dieser Bandbreite 780kBit/s zur Verfügung. Bei Version 2.0 wurde diese Bandbreite durch neue Modulationsverfahren auf 2178 kBit/s erhöht. Da Bluetooth das ISM-Band mit einer Reihe andere Anwendungen teilen muss, wird der Übertragungskanal in Zeitslots aufgeteilt und ein Frequenzsprung-Verfahren (Frequency Hopping Spread Spektrum) angewandt. Dabei wird in kurzer Folge die Frequenz gewechselt. Der Wechsel erfolgt jeweils nach jedem Paket (625 μs). Dies soll eine gegenseitige Behinderung von Anwendungen im ISM-Band oder die Behinderung andere Bluetooth-Verbindungen verhindern.

Um die Kommunikation mehrerer Geräte zu ermöglichen, baut Bluetooth auf ein Master/Slave Konzept. Das erste Endgerät, welches einen Verbindungsaufbau startet, wird automatisch zum Master. Somit kann jedes Endgerät die Rolle des Masters oder eines Slaves übernehmen. Die Hardwareadresse des Masters bestimmt die Abfolge der Frequenzsprünge. Ein von einem Endgerät erstelltes Netz wird als Piconetz bezeichnet. In einem Piconetz sind 1 Master und maximal 7 Slaves erlaubt. Über die Verteilung der Zeitslots im Piconetz entscheidet der Master. In einem Piconetz ist es nur dem Master möglich mit allen Slaves zu kommunizieren. Die Slaves können nur mit dem Master kommunizieren. Daher wurde eine Möglichkeit den Master zu wechseln vorgesehen (Master Slave Role Switch) (vgl. [43]).

Die Sendeleistung von Bluetooth Geräten wird in drei Klassen eingeteilt. Geräte der Leistungsklasse 1 senden mit einer Leistung von bis zu 100mW und können eine Verbindung auf bis zu 100m herstellen. Geräte der Sendeklasse 2 senden mit 2,5mW und Geräte der Sendeklasse 1 mit maximal 1mW. Diese Geräte sind auf einen Kommunikationsbereich von etwa 10m beschränkt (vgl. [43]).

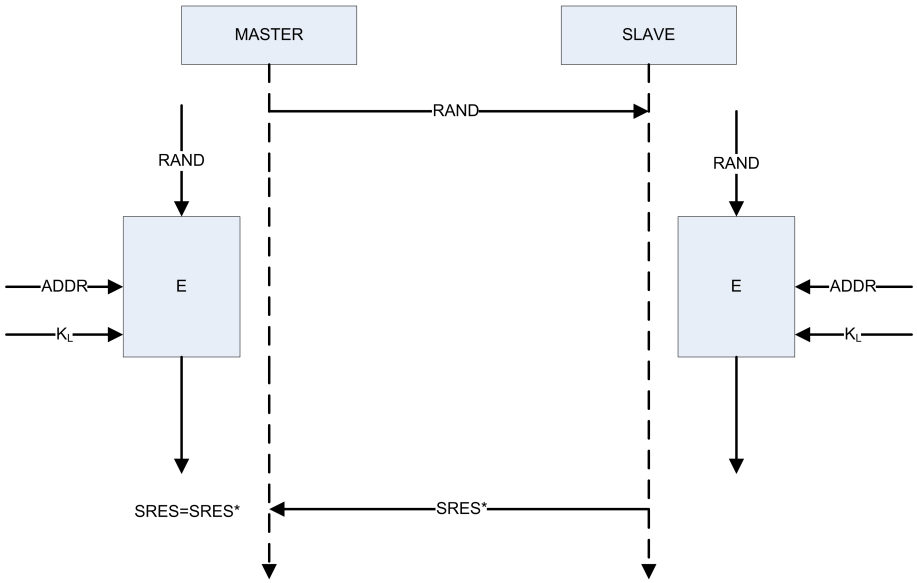

Abbildung 1.5: Authentifizierung bei einer Bluetooth-Verbindung [43]

Auch im Bereich der Sicherheit gibt es eine Einteilung in Klassen. Die Wahl der Sicherheitsklasse wird dabei der kommunizierenden Anwendung überlassen. So wird im Security Mode 1 keine Verschlüsselung und Authentifizierung durchgeführt. Dieser Modus eignet sich daher für weniger sicherheitsrelevante Anwendungen, wie zum Beispiel den Austausch von elektronischen Visitenkarten. Bei Security Mode 2 wird die Entscheidung über die verwendeten Sicherheitsmerkmale dem Benutzer überlassen. Security Mode 3 dagegen erfordert explizit eine Authentifizierung und Verschlüsselung. Dies basiert auf einem zuvor durchgeführten Pairing. Beim Pairing fordert jedes Endgerät die Eingabe eines Pins. Aus diesem PIN wird zwischen beiden Geräten ein Link Key K_L generiert und gespeichert. Der Link Key dient in weiterer Folge zur Authentifizierung und dem Aufbau einer gesicherten Verbindung. Bei der Authentifizierung erzeugt das initiierende Gerät eine Zufallszahl ($RAND$) und überträgt diese an den Kommunikationspartner. Dieser generiert aus der Zufallszahl, dem Link Key und der Bluetooth Adresse des Masters ($ADDR$) eine Antwort (Signed Response $SRES$) für den Master. Dieser vergleicht die erhaltene Antwort mit der von ihm selbst generierten Antwort (vgl. [43]). Dieses Verfahren wird in Abbildung 1.5 dargestellt.

Zur Verschlüsselung der Verbindung wird ein eigener Ciphering Key K_C generiert. Dieser wird aus dem Link Key und einer Zufallszahl gebildet. Die Zufallszahl muss für jede Verbindung neu generiert werden. Das soll sicherstellen, dass bei jeder Verbindung ein anderer Schlüssel zum Einsatz kommt. Der Ciphering Key bildet mit der Echzeituhr und der Bluetooth Adresse des Masters die Eingangswerte für den SAFER+ Algorithmus. Bei SAFER+ handelt es sich um einen von der ETH Zürich entwickelten Stromchiffre (vgl. [43]). Dargestellt wird dieses Verfahren in Abbildung 1.6.

Um die Interoperabilität zwischen verschiedenen Endgeräten zu gewährleisten, sind im Bluetooth-Standard Profile definiert. Diese Profile basieren auf einem Server/Client Prinzip und stellen Funktionalität für verschiedenste Anwendungen zur Verfügung. Es existieren zum Beispiel Profile für Datenübertragung, Adress-/Telefonbuchsynchronisation, für serielle Verbindungen, Dial-Up Verbindungen und zum Telefonieren mit Freisprecheinrichtungen (vgl. [43]).

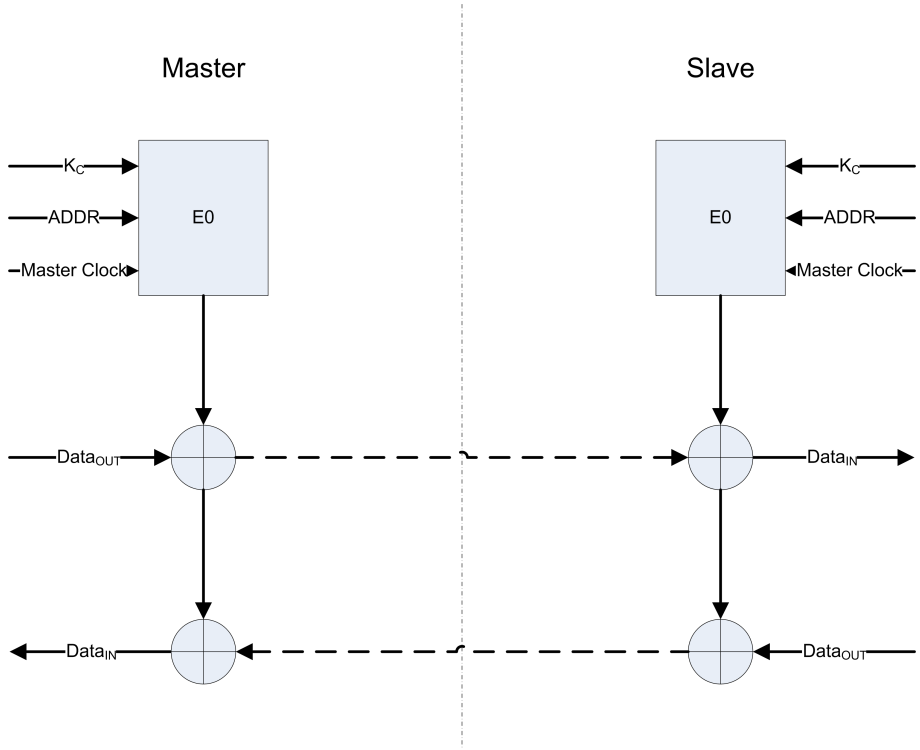

Abbildung 1.6: Verschlüsselung durch SAFER+ [43]

1.4 Datenschutz und Anonymisierung

Datenschutz gewinnt in einer vollkommen vernetzten Welt immer mehr an Bedeutung. Ein Mittel, den Datenschutz sicherzustellen aber auf das Sammeln von Daten nicht zu verzichten, ist die Anonymisierung aller personenbezogenen Daten. Den Begriff Anonymisierung definiert das deutsche Ministerium für Justiz im Datenschutzgesetz 2009 §3 (vgl. [22]) wie folgt:

> "§3 Weitere Begriffsbestimmungen:
> ....
> (6) Anonymisieren ist das Verändern personenbezogener Daten derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können."

Ein Problem der Anonymisierung ist die Auswahl der Bereiche, die einer Anonymisierung unterzogen werden sollen. Oft ist es nicht ausreichend, nur die eindeutigen Identifikatoren wie Name oder Sozialversicherungsnummer aus dem Datensatz zu entfernen. Sind noch Daten wie Wohngebiet, Arbeitgeber oder die Farbe des Autos vorhanden, kann durch eine Verknüpfung dieser Merkmale leicht auf die betreffende Person rückgeschlossen werden.

> "(6a) Pseudonymisieren ist das Ersetzen des Namens und anderer Identifikationsmerkmale durch ein Kennzeichen zu dem Zweck, die Bestimmung des Betroffenen auszuschließen oder wesentlich zu erschweren."

In diesem speziellen Fall der Anonymisierung, wird die Identität des Betroffenen nicht aus dem Datensatz gelöscht. Sie wird durch ein Pseudonym ersetzt. Pseudonyme sind für die Internetgeneration schon lange ein Mittel um ihr Spuren im Internet zu verschleiern. Dazu werden reale Namen durch sogenannte "Nicknames" ersetzt. Ein ähnliches Verfahren wird auch im Bereich der datenschutzkonformen Datenerhebung durchgeführt. In diesem Fall wird der Identifikator im Datensatz durch einen mit kryptographischen Mitteln errechneten Wert ersetzt. Grund für diese, im Gegensatz zur Entfernung aller personenbezogenen Daten wesentlich aufwendigeren Methode, ist die Möglichkeit die Zuordnung von verschieden Quellen akquirierten Daten zu gewährleisten. Wird der neue Identifikator von allen Quellen mit dem selben Algorithmus errechnet, können Daten aus unterschiedlichen Quellen zusammengefasst werden und eine Aussage über die Person wiedergeben, ohne die Person selbst preiszugeben.

Verkehrsflussanalysen sind ein Bereich in dem Pseudonymisierung eingesetzt wird. Verschiedene Quellen erfassen die Fahrzeuge. Aus diesen Daten soll ein Muster des Verkehrsflusses erstellt werden. Daher ist es unumgänglich, dass ein Fahrzeug bei allen Erfassungspunkten den selben Identifikator erhält. Jedoch dürfen keine Daten, die für eine Nachverfolgung der Person geeignet wären, verwendet werden um konform mit dem Datenschutzgesetz zu sein (vgl. [1]).

Im speziellen Fall der Vorratsdatenspeicherung ist eine solche Anonymisierung jedoch von Seiten des Gesetzgebers nicht erwünscht. Eine EU-Richtlinie legt fest, welche Daten von Telekommunikationsanbietern zu speichern sind. Zu diesen Daten zählen zum Beispiel die Benutzerkennung, Datum, Uhrzeit und Standortdaten. Dies wiederspricht dem Hang zur totalen Anonymisierung benutzerbezogener Daten (vgl. [50]). Eine solche Richtlinie existiert bislang nur für Anbieter von Telekomunikationsdiensten. Wird das Potential von Verkehrserfassungssystemen von den zuständigen Gremien erkannt, ist eine Portierung einer solchen Richtlinie jedoch möglich.

Kapitel 2

Kryptographie

Seit Anbeginn der Geschichtsschreibung verwenden Menschen Kryptographie um ihr Wissen und ihre Informationen vor Dritten versteckt zu halten. Nicht immer erfolgreich und anfangs mit primitiven Mitteln wurde versucht, dem Feind den Blick auf die eigenen Pläne zu verwehren, Geheimnisse zu verstecken oder Nachrichten sicher zu übertragen. Seither wurde die Entwicklung in diesem Bereich unermüdlich vorangetrieben. Erst der Einsatz moderner elektronischer Rechenmaschinen machte Kryptographie in der heutigen Form möglich. War Kryptographie früher die reine Ver- und Entschlüsselung von Daten, ist dieses Gebiet heute wesentlich umfangreicher. Durch die Menge an Informationen die unseren Alltag beherrscht, ist Kryptographie heute allgegenwärtig. Die moderne Kryptographie stützt sich auf 4 Prinzipien (vgl. [36]):

- Vertraulichkeit (Confidentiality)
- Integrität (Integrity)
- Authentizität (Authenticity)
- Verbindlichkeit (Non repudiation)

Vertraulichkeit: Für Dritte ist es nicht möglich, die Information in der Nachricht zu erkennen.

Integrität: Eine Nachricht kann nicht von Dritten geändert werden, ohne dass dies erkannt wird.

Authentifizierung: Es ist für Dritte nicht möglich, sich als Kommunikationspartner auszugeben und somit die Gegenstelle zu täuschen.

Verbindlichkeit: Ein Abschicken der Nachricht kann nicht abgestritten werden.

Für die Ver- und Entschlüsselungsfunktionen wird angenommen, dass diese allgemein bekannt sind. Somit hängt die Stärke der Verschlüsselung nicht von der Geheimhaltung der verwendeten Funktion, sondern von der Stärke der Funktion an sich und der Geheimhaltung des Schlüssels ab. Dies hat August Kerckhoff bereits 1883 definiert. Seine Annahme ist ein Grundsatz der modernen Kryptographie (vgl. [47])

Für eine gemeinsame Gesprächsbasis über dieses vielschichtige Thema, ist es notwendig anfangs eine Bestimmung der verwendeten Begriffe durchzuführen. Die wichtigsten dieser Definitionen werden im Folgenden aufgelistet:
Unter **Verschlüsselung** versteht man die Umwandlung von Klartext in eine chiffrierte Form unter Zuhilfenahme eines geheimen Schlüssels (vgl. [44]):

$$c = E(m, k)$$

oder

$$c = E_k(m)$$

- m bezeichnet den Klartext/die zu verschlüsselnde Nachricht (Message)
- c bezeichnet den verschlüsselten Text (Ciphertext)
- k bezeichnet den geheimen Schlüssel, der zur Verschlüsselung verwendet wird (Key)
- E bezeichnet die Verschlüsselungsfunktion (Encryption algorithm)

Als **Entschlüsselung** wird der umgekehrte Prozess bezeichnet. Wobei hier D die verwendete Entschlüsselungsfunktion darstellt (Decryption algorithm) (vgl. [44]):

$$m = D(c, k)$$

oder

$$m = D_k(c)$$

Bei diesem Beispiel ist es notwendig, dass beide Parteien über den Schlüssel k verfügen, um miteinander kommunizieren zu können. Daher ist die Geheimhaltung dieses Schlüssels unbedingt sicher zu stellen. Bei kompromittiertem Schlüssel könnte ein Angreifer die gesamte Kommunikation mitverfolgen. Solche Verschlüsselungssysteme werden in weiterer Folge als **Symmetrische Verschlüsselung** bezeichnet. Auf diese Form der Verschlüsselung wird in Kapitel 2.1 näher eingegangen.

Eine andere Form der Kryptographie benutzt verschiedene Schlüssel, die über eine mathematische Verbindung verfügen. Der öffentliche Schlüssel wird zum Verschlüsseln verwendet, der private Schlüssel zum Entschlüsseln. Dies wird als **Asymmetrische Verschlüsselung** oder **Public Key Kryptographie** bezeichnet (vgl. [44]). Näheres dazu in Kapitel 2.2.

Sicherheit kann aber nicht nur durch einen starken kryptographischen Algorithmus erreicht werden. Ein wichtiger Faktor ist der richtige Einsatz des Algorithmus. Diesem Thema widmet sich das Kapitel 2.5. Dazu hat sich in der Fachliteratur eingebürgert, die bei einer Kommunikation beteiligten Parteien mit folgenden Namen zu bezeichnen (vgl. [44]):

- Alice
- Bob
- Eve (meist der Angreifer)

Dazu jedoch mehr in Kapitel 2.5. Im nun folgenden Kapitel werden in Kürze die Anfänge der Kryptographie behandelt.

2.1 Symmetrische Kryptographie

Symmetrische Kryptographie bezeichnet alle Algorithmen bei denen die Ver- und Entschlüsselung mit dem selben Schlüssel durchgeführt wird. Wenn ein Schlüssel einfach vom anderen abgeleitet wird, bezeichnet man dies ebenfalls als symmetrische Kryptographie. Das Feld der symmetrischen Verschlüsselung teilt sich dabei in zwei Bereiche. Blockchiffren bearbeiten einen Block von Daten (Bytes oder mehrere Buchstaben), wohingegen bei einer Stromchiffre immer eine Einheit bearbeitet wird (Bits, ein Buchstabe). Jedes dieser zwei Verfahren hat spezifische Eigenheiten auf die im folgenden Abschnitt näher eingegangen wird (vgl. [44]):

2.1.1 Stromchiffren (Streamcipher)

Stromchiffren (Streamcipher) bearbeiten pro Arbeitsschritt immer eine Teileinheit der Daten und sind dabei meist sehr schnell. Dazu wird meist ein Schlüsselstrom mit einem Datenstrom per XOR verknüpft (siehe Abbildung 2.1).

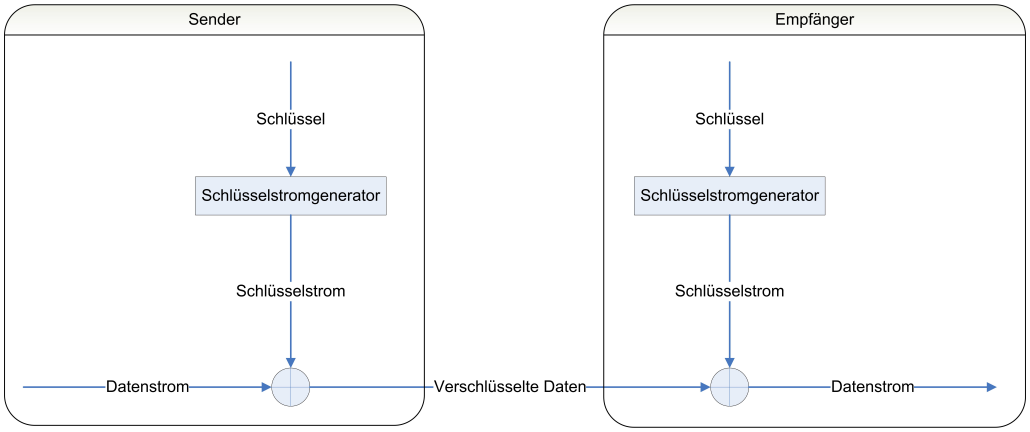

Abbildung 2.1: Stromchiffren

Stromchiffren sind eine sehr schnelle Möglichkeit zur Verschlüsselung. Sie leiden aber teilweise an Sicherheitsproblemen, deren Lösung einen häufigen Schlüsseltausch notwendig machen. Näheres zu Stromchiffren findet man im Buch "Cryptography - An Introduction" [44].

2.1.2 Blockchiffren (Blockcipher)

Blockchiffren (Blockcipher) bearbeiten pro Arbeitsschritt einen Block an Daten (Bytes oder mehrere Buchstaben). Dazu wird kein interner Zustand gespeichert und jeder Block wird eigenständig bearbeitet (siehe Abbildung 2.2). Es ist aber dennoch möglich, den vorigen Block in die Verarbeitung einzubeziehen. Dazu werden die Blockchiffren in verschiedenen Verarbeitungsmodi betrieben.

Blockchiffren sind meist etwas langsamer als Stromchiffren, zeichnen sich aber im Gegensatz zur asymmetrischen Kryptographie immer noch durch eine sehr hohe Verarbeitungsgeschwindigkeit aus. Oft wird bei einer Blockchiffre eine einfache Funktion auf die Daten angewandt, dies aber mehrmals hintereinander. Bei jeder dieser Runden kommt ein anderer Rundenschlüssel zum Einsatz, der aus dem Originalschlüssel abgeleitet wird.

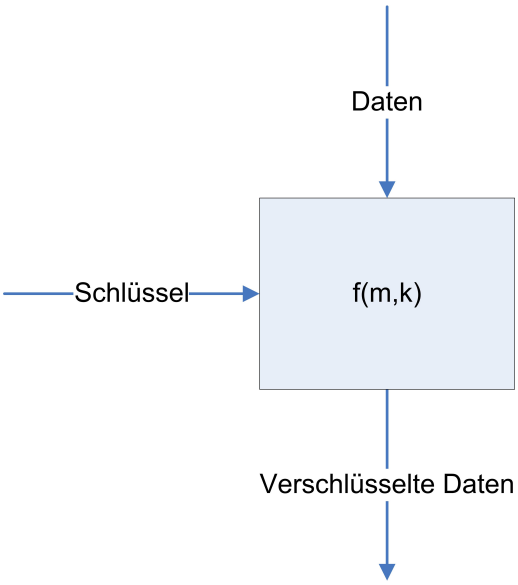

Abbildung 2.2: Blockchiffren

Blockchiffren können in zwei Gruppen aufgeteilt werden. Zum einen existieren Feistel-Netzwerke, die auf einer von Horst Feistel entwickelten Struktur basieren (siehe Abbildung 2.3) (vgl. [46]).

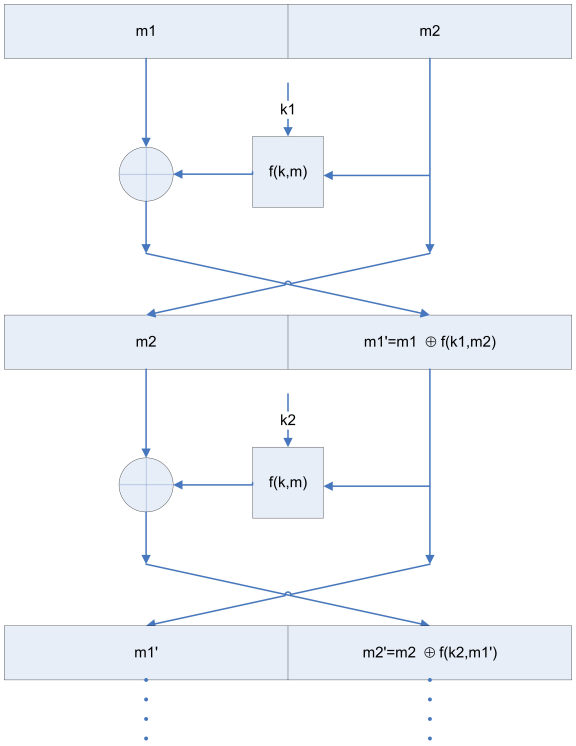

Abbildung 2.3: Feistel Netzwerk (3 Runden)

Die Verschlüsselung funktioniert bei Feistel-Netzwerken meist gleich wie die Entschlüsse-

lung. Es müssen nur die Rundenschlüssel vertauscht werden. Ein prominentes Beispiel für ein Feistel-Netzwerk ist der Data Encrytion Algorithmus (DES). Er fand in zahlreichen Applikationen zur Verschlüsselung großer Datenmengen Anwendung. Näheres zu DES und Feistel Netzwerken findet man im Buch Cryptography and Network Security (vgl. [46]).

Die andere Art von Blockchiffren wird als Substitutions-Permutations-Netzwerk (SP-Netzwerk) bezeichnet. Ein solches Netzwerk führt in mehreren Runden verschiedene Substitutions- und Permutationsschritte durch, um die Rohdaten zu verschlüsseln. Diese Art von Blockchiffren wird in Sektion 2.1.4 anhand des Advanced Encryption Standard näher beschrieben. Näheres zu SP-Netzwerken findet man unter [47].

2.1.3 Modes of Operation

Sollen Daten, die größer als die Blocklänge des Algorithmus sind, verschlüsselt werden, müssen die Daten in Blöcke aufgeteilt werden. Die Daten (M) werden dabei als eine Kette von Bytes betrachtet. Diese Kette wird in n Blöcke, die jeweils der Blocklänge des Algorithmus entsprechen, aufgeteilt. Ist die Länge der Daten kein ganzzahliges Vielfaches der Blocklänge, so werden die Daten mit einem Padding p auf die benötigte Länge gebracht.

$$M = m_1 \parallel m_2 \parallel \cdots \parallel m_n \parallel (p)$$

Die Blöcke werden dann anhand definierter Verfahrensweisen (Modes of Operation) verschlüsselt. Nach der Verschlüsselung wird die verschlüsselte Nachricht C aus den einzelnen Ciphertextblöcken c_i zusammengesetzt:

$$C = c_1 \parallel c_2 \parallel \cdots \parallel c_n$$

2.1.3.1 Electronic Codebook Mode

Das einfachste dieser Verfahren ist der sogenannte **Electronic Codebook Mode (ECB)**. Jeder Datenblock wird verschlüsselt und führt zu einem Ciphertextblock. Außer dem Schlüssel fließen keine anderen Komponenten in die Verschlüsselung ein. Ebenso werden keine anderen Datenblöcke oder Ciphertextblöcke berücksichtigt (vgl. [46]).

Verschlüsselung und Entschlüsselung mit ECB:

$$1 \leq i \leq n$$

$$c_i = E(m_i, k)$$

$$m_i = D(c_i, k)$$

Ein Vorteil des Electronic Code Book Modes ist die einfache Struktur. Zur Umsetzung wird dabei nur die Verschlüsselungsfunktion ohne weitere Verknüpfungen benötigt. Zusätzlich ist die parallele Verarbeitung aller Datenpakete möglich. Dies führt zu hoher Verarbeitungsgeschwindigkeit bei optimierten Implementierungen in Hardware. Weiters betrifft ein Fehler in einem Chiffreblock immer nur einen Klartextblock. Fehler werden nicht an andere Blöcke übertragen. Viel schwerer für den praktischen Einsatz wiegen jedoch die Nachteile des Electronic Code Book Modes. So sind Strukturen in den Daten auch nach einer Verschlüsselung sichtbar. Dies ist nicht zu verhindern, da bei ECB jeder Klartextblock mit gleichem Inhalt auf einen Chiffreblock mit gleichem Verschlüsselungsergebnis abgebildet wird. Weiters können Chiffreblöcke einfach ausgetauscht werden. Der Austausch betrifft dann bei der Entschlüsselung auch nur diesen einen Block. Betrifft dies

Klartextteile ohne vorgegebene Struktur (zum Beispiel Schlüsselmaterial oder Zufallszahlen)und werden keine weiteren Sicherheitsmaßnahmen, wie etwa Modification Detection Codes (MDC), angewandt, können diess auf einfache Weise durch einen Angreifer ausgetauscht werden. Diese Eigenschaften führen zu massiven Schwächen. ECB wird daher immer seltener eingesetzt (vgl. [46, 19]).

2.1.3.2 Cipher Block Chaining Mode

Als Alternative bietet sich der **Cipher Block Chaining Mode (CBC)** an. Dabei wird das Ergebnis der Verschlüsselung des Blocks c_i mit dem nächsten Datenblock m_{i+1} verschlüsselt. Der erste Block wird mit einem Initialwert (Initial Vector, IV) verknüpft (siehe Abbildung 2.4) (vgl. [19]).

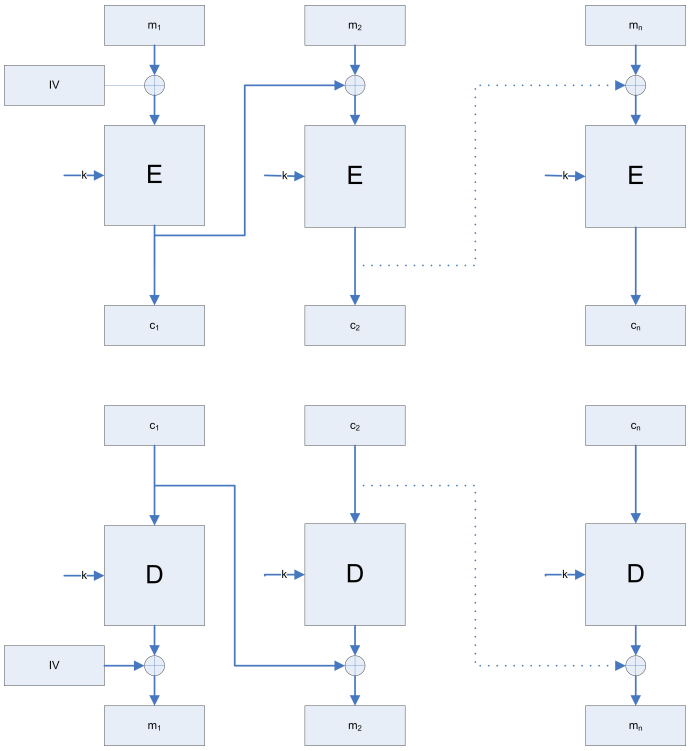

Abbildung 2.4: Block Cipher Chaining Mode [19]

$$2 \leq i \leq n$$

$$c_1 = E((m_1 \oplus IV), k)$$

$$c_i = E((m_i \oplus c_{i-1}), k)$$

$$m_1 = IV \oplus E(c_1, k)$$

$$m_i = c_{i-1} \oplus E(c_i, k)$$

Durch die Einbindung von unterschiedlichen Ininitialwerten wird erreicht, dass bei mehrmaliger Verschlüsselung des selben Klartextes unterschiedliche Ciphertexte entstehen. Strukturen werden bei CBC durch die Verkettung der Verschlüsselung der einzelnen

Blöcke erreicht. Des Weiteren ist es bei CBC nicht möglich einzelne Chiffretextblöcke auszutauschen. Eine Änderung in einem Chiffretextblock wirkt sich immer auf einen weiteren Block aus. Dies ist im Bereich der Fehlertoleranz ein Nachteil. Durch diese Eigenschaft ist es auch nicht möglich mehrere Blöcke parallel zu verarbeiten. Auch bei Implementationen in Hardware kann daher durch parallele Verarbeitung kein Geschwindigkeitsvorteil erzielt werden (vgl. [46, 19]).

Der CBC kann auf einfache Weise zur Erstellung eines Message Authentication Codes (MAC, mehr dazu in Abschnitt 2.3.3) verwendet werden. Dazu wird nur der letzte Chiffretextblock berücksichtigt (C_n). Alle weiteren Chiffretextblöcke werden verworfen. Der letzte Block stellt den MAC des Klartextes dar (vgl. [46, 19]).

2.1.3.3 Counter Mode

Abschließend exisiert noch die Variante als **Counter Mode**. Dabei wird ein Zählwert für jeden Block erhöht und verschlüsselt. Das Ergebnis wird mit dem Datenblock XOR verknüpft und ergibt den Ciphertextblock. Dabei ist eine parallele Verabeitung möglich. Diese Variante wird in Abbildung 2.5 dargestellt (vgl. [19]).

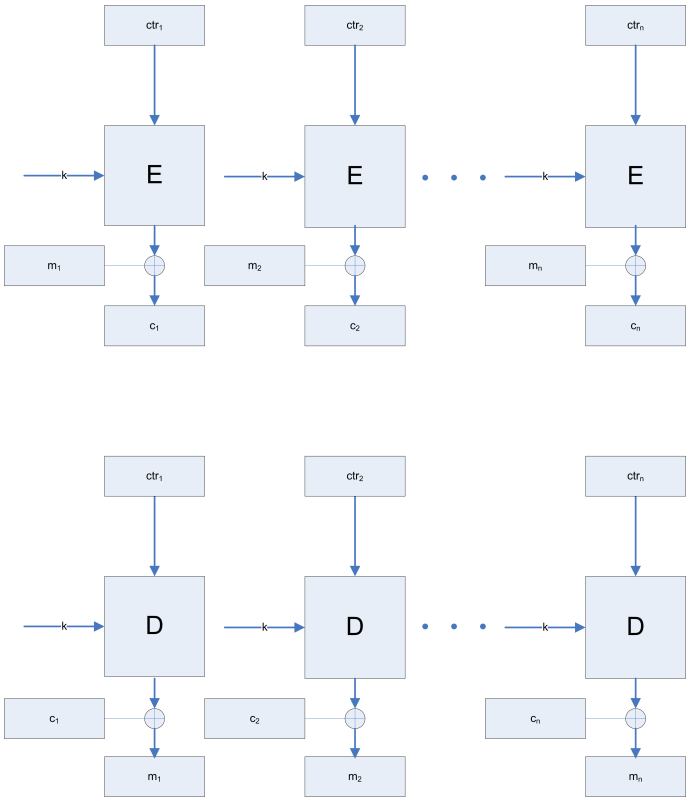

Abbildung 2.5: Counter Mode [19]

$$y_1 = E(ctr, k)$$

$$1 \leq i \leq n$$

$$y_i = E(ctr + i, k)$$

$$c_i = m_i \oplus y_i$$

$$m_i = c_i \oplus y_i$$

Unterschiedliche Startwerte (*ctr*) führen bei gleichem Klartext zu unterschiedlichen Chiffretexten. Jedoch muss dieser Startwert auch bei jeder Verschlüsselung anders und weit genug entfernt von vorigen Startwerten gewählt werden. Würden zwei Nachrichten mit dem selben Klartext verschlüsselt, könnte ein Angreifer durch die Kombination der beiden Chiffretexte auf den Inhalt einer Nachricht rückschließen. Eine Implementation der Entschlüsselung ist nicht notwendig, da zur Entschlüsselung einfach nochmals der verschlüsselte Zählwert verknüpft wird. Da sich Entschlüsselung und Verschlüsselung mit Ausnahme der zu verabeitenden Daten nicht unterscheiden, ist auch hier nur eine Implementation notwendig. Die verschlüsselten Zählwerte können zudem weit im Voraus berechnet und gespeichert werden. Zur Verschlüsselung müssen dann nur noch die Klartextblöcke mit den bereits gespeicherten Werten per XOR verknüpft werden. In Hardware implementiert, kann durch parallele Verarbeitung der Datenblöcke ein hoher Geschwindigkeitsvorteil erzielt werden. Ein Fehler im Chiffretext beschränkt sich nur auf einen Datenblock und wird nicht an benachtbarte Blöcke weitergegeben (vgl. [46, 19]).

2.1.4 Advanced Encrytption Standard

Der Advanced Encryption Standard (AES) wurde als Nachfolger für den in die Jahre gekommenen Data Encryption Standard gesucht. Dazu wurden in einem offenen Bewerb verschiedene symmetrische Verfahren verglichen. Der von Joan Daemen und Vincent Rijmen entwickelte Algorithmus Rijndael konnte sich gegen alle Mitbewerber durchsetzen und wurde im Oktober 2000 als Advanced Encryption Standard bekanntgegeben. Der Algorithmus ist ein SP-Netzwerk und kann mit verschiedenen Schlüssellängen betrieben werden (128, 192 oder 256 Bit). Die Länge des Datenblocks ist auf 128 Bit festgelegt (Rijndael an sich unterstützt verschiedene Blocklängen). Zur Verschlüsselung wird jeder Datenblock in mehreren Runden verschiedenen Transformationen unterzogen. Zur Repräsentation wird der Schlüssel und jeder Datenblock in eine Tabelle geschrieben. Diese Tabelle besteht aus 4 Zeilen, wobei jedes Feld ein Byte darstellt. Die Anzahl der Spalten wird bei der Repräsentation für den Schlüssel durch die Schlüssellänge bestimmt. Beim Datenblock ist diese durch die vorgegebene Blocklänge von 128 Bit auf 4 Spalten festgelegt (siehe Abbildung 2.6) (vgl. [32]).

Abbildung 2.6: Repräsentation von Daten und Schlüssel bei AES [32]

Die Repräsentation der Daten wird in weiterer Folge als *State* bezeichnet. *State* wird nun in einer Anzahl von Runden einer Transformation unterzogen. Die Anzahl der Runden

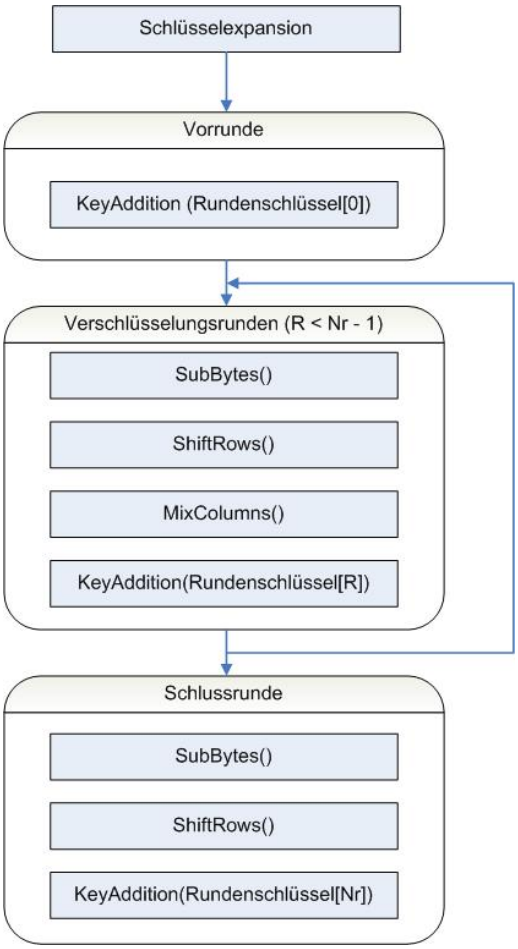

Abbildung 2.7: Ablauf einer AES Verschlüsselung [32]

(Nr) wird durch die Schlüssellänge bestimmt. Bei einer Schlüssellänge von 128 Bit müssen 10 Runden durchgeführt werden (12 Runden bei 192 Bit und 14 Runden bei 256 Bit). Die verschiedenen Transformationen bei der Verschlüsselung eines Datenblocks werden in Abbildung 2.7 dargestellt und im Folgenden genauer erläutert (vgl. [32]).

S-Box Ein wichtiger Bestandteil des AES ist die Substitutionsbox (S-Box). Diese ist in einer Tabelle implementiert (siehe Abbilung 2.8)

Dabei wird jedem Byte (repräsentiert als xy in hexadezimaler Schreibweise) ein Ausgangswert zugewiesen. Das Byte 0xA8 würde also durch 0xC2 ersetzt. Die S-Box bei AES ist invertierbar und basiert im Gegensatz zu anderen Algorithmen auf einer mathematischen Beschreibung (näheres zum Design der S-Box in der Spezifikation zu AES (vgl. [32]) und im Vorschlag zum AES Bewerb (vgl. [16])).

Schlüsselexpansion Anfangs müssen aus dem Hauptschlüssel eine Anzahl ($Nr + 1$) von Rundenschlüsseln erzeugt werden. Dazu wird der Schlüssel in Wörter aufgeteilt (ein Wort jeweils 4 Byte, entspricht einer Spalte in der tabellarischen Repräsentation). Jeder Rundenschlüssel muss genau die gleiche Länge wie ein Datenblock haben. Um dies zu erreichen, wird der originale Schlüssel an den Anfang einer Tabelle mit 4 Zeilen und

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Abbildung 2.8: AES S-Box [32]

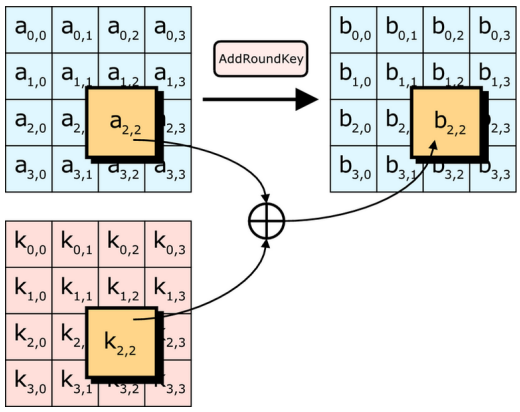

Abbildung 2.9: AES AddRoundKey [15]

$(Nr + 1) \cdot 4$ Spalten geschrieben. Die Werte in den weiteren Spalten errechnen sich wie folgt (vgl. [32]):

$$k_{i,j} = \begin{cases} k_{i-6,j} + S(k_{i-1,j-1}), & \text{wenn } i \equiv 0 \pmod 6, \\ k_{i-6,j} + k_{i-1,j}), & \text{ansonsten.} \end{cases}$$

Ein Vorteil dieser Schlüsselexpansion ist, dass sie erst zur Laufzeit erfolgen kann und somit Speicherplatz eingespart wird.

AddRoundKey In der Funktion AddRoundKey wird jedes Byte von *State* mit dem zugehörigen Byte des Rundenschlüssels per XOR verknüpft. Dies ist die einzige Funktion, die den Schlüssel in die Verschlüsselung einbringt. Die Funktion wird in Abbildung 2.9 grafisch erklärt (vgl. [32]).

SubBytes Die Funktion SubBytes ersetzt jedes Byte in *State* durch ein Äquivalent. Dazu wird eine S-Box verwendet. Dies entspricht einer monoalphabetischen Verschlüsselung (Jeder Klartextbuchstabe wird genau einem Geheimtextbuchstaben zugeordnet) (vgl. [32]). Die Funktion SubBytes wird in Abbildung 2.10 näher dargestellt.

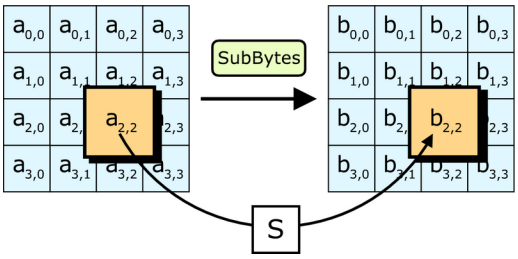

Abbildung 2.10: AES SubBytes [15]

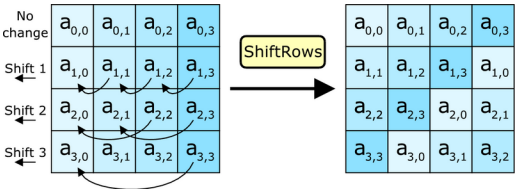

Abbildung 2.11: AES ShiftRows [15]

ShiftRows Die Funktion ShiftRows rotiert jede Zeile nach links. Die Rotation ist abhängig vom Index der Zeile. So wird die Zeile null nicht rotiert. Die Zeile eins wird um eine Zelle nach links rotiert. (Zeile zwei und drei jeweils um zwei und drei Zellen). Dies schützt gegen bestimmte Attacken (näheres dazu im Vorschlag zum AES Bewerb (vgl. [16])). Die Funktion wird in Abbildung 2.11 dargestellt.

MixColumns Bei MixColumns wird jede Spalte von *State* mit einer bestimmten Matrix multipliziert. Dies führt zu einer Vermischung aller Werte in dieser Spalte. Besonderes Augenmerk bei der Wahl der Matrix wurde auf Invertierbarkeit, Symmetrie, gute Durchmischung und eine einfache Beschreibung gelegt (vgl. [16]). Im Folgenden wird die Funktion in Abbildung 2.12 dargestellt:

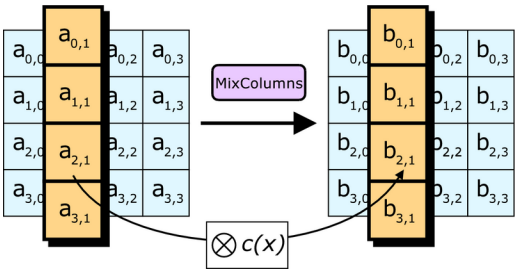

Abbildung 2.12: AES MixColumns [15]

Analyse des AES Der Advanced Encryption Standard setzt alle Komponenten eines SP-Netzwerkes um. Zusätzlich zur Vermischung mit dem Schlüssel (AddRoundkey), der Substitution (SubBytes) und Permutation (ShiftRows), wird eine zusätzliche lineare Transformation durchgeführt (MixColumns)(vgl. [47]). Im Folgenden werden einige Eigenschaften des AES aufgelistet:

- Die erste und letzte Operation ist eine Vermischung mit einem Rundenschlüssel. Dies

wird als Whitening bezeichnet und soll den Input zur ersten Verschlüsselungsrunde und den Output nach der letzten Runde verschleiern (vgl. [47]).

- Die besondere Wahl der S-Box wirkt einer linearen und differentialen Cryptoanalyse entgegegen (vgl. [16]).
- Alle Funktionen sind invertierbar und eine Entschlüsselung ist dadurch einfach möglich (vgl. [32])
- Beim Entwurf des Algorithmus wurde besonderes Augenmerk auf Geschwindigkeit und Einfachheit gelegt (vgl. [16]).

Mögliche Angriffe Einige bis jetzt bekannte Angriffe zielen auf spezifische Implementationen des Standards ab (Side-Channel-Angriffe). Ein Beispiel dafür sind Angriffe durch Informationen, die über den Cache eines Prozessors gewonnen werden (vgl. [35]) oder Angriffe die durch Einbringen eines Fehlers in die Verschlüsselung zum Ziel führen (vgl. [41]). Zusätzlich sind Attacken auf eine reduzierte Version des AES mit weniger Runden[4] und Angriffe mit schwachen Schlüsseln bekannt (vgl. [5]). Im Jahr 2009 wurden weitere Attacken, die sich verwandte Schlüssel zur Hilfe nehmen, bekannt (vgl. [7, 6]). Diese Angriffe sind zur Zeit jedoch praktisch noch keine Gefahr.

2.1.5 Zusammenfassung

Grundsätzlich ist zu sagen, dass sich symmetrische Chiffren gut zur Verschlüsselung großer Datenmengen eignen, da sie im Gegensatz zu asymmetrischer Kryptographie wesentlich höhere Verarbeitungsgeschwindigkeiten erzielen. Ein Einsatzgebiet bei BLIDS wäre die Übertragung von Daten der Sensorstationen an den Datenbankserver. Zu lösen ist jedoch noch das Problem des Schlüsselmanagements (dazu mehr in Abschnitt 2.2 und Abschnitt 2.5).

2.2 Asymmetrische Kryptographie

Mit dem Advanced Encryption Standard steht uns eine Möglichkeit zur Verfügung, große Datenmengen auf sicherem Weg zwischen zwei Kommunikationsteilnehmern zu übermitteln. Weiterhin ungelöst ist das Problem, wie beide Teilnehmer an den geheimen Schlüssel kommen sollen. Außerdem sollte bei der vermehrten Übertragung von Daten der geheime Schlüssel häufig gewechselt werden. Dies erhöht die Schwierigkeit für einen Angreifer. Zusätzlich wünscht man sich ein Instrument, um die Identität des Gegenübers sicher feststellen zu können. Für all diese Probleme kann asymmetrische Kryptographie eine mögliche Lösung darstellen.

Bei asymmetrischer Kryptographie existieren zwei Schlüssel. Der Eine wird als öffentlicher Schlüssel bezeichnet und ist allgemein zugänglich. Den Anderen nennt man privater Schlüssel. Dieser Schlüssel muss unter allen Umständen nur dem Eigentümer bekannt sein. Die beiden Schlüssel sind über eine mathematische Funktion verbunden. Vorgestellt wurde ein solches Konzept erstmals 1975 durch Diffie und Hellman. Diese mathematische Funktion sollte folgende Eigenschaften besitzen (vgl. [44]):

- Aus dem einen Schlüssel kann auf einfache Weise der zweite Schlüssel generiert werden.

- Umgekehrt soll es unmöglich oder besonders schwer sein vom zweiten Schlüssel auf den ersten zu schließen, wenn zusätzliche Information nicht bekannt ist.

Eine solche Funktion wird als Trapdoor-Einwegfunktion bezeichnet. Eine Anzahl solcher Einwegfunktionen ist seit längerem bekannt und gut erforscht. Im Folgenden werden zwei gut bekannte Einwegfunktionen in der Kryptographie (vgl. [44]) vorgestellt:

- Die Faktorisierung großer zusammengesetzter Zahlen
- Berechnung des diskreten Logarithmus

Näheres zu den mathematischen Grundlagen findet man in "Cryptography - An Introduction" ([44]). In den folgenden Abschnitten werden nun zwei asymmetrische Verfahren näher beschrieben.

2.2.1 RSA

Der RSA Algorithmus wurde von Rivest, Shamir und Adleman entwickelt und gilt als erfolgreichstes asymmetrisches Verfahren. Die Sicherheit von RSA basiert auf der Faktorisierung großer zusammengesetzter Zahlen. Im Folgenden wird der Algorithmus kurz beschrieben:

Wähle 2 große Primzahlen p und q. Berechne

$$N = p \cdot q$$

und

$$\varphi(N) = (p-1) \cdot (q-1).$$

Wähle e so, dass gilt

$$gcd(e, \varphi(N) = 1.$$

Meist wird für e 3, 17 oder 65537 gewählt. Berechne d

$$d \cdot e \equiv 1 \pmod{\varphi(N)}.$$

Der öffentliche Schlüssel besteht nun aus (N, e) und der private aus (p, q, d). Um eine geheime Nachricht an jemanden zu schicken reicht es, diese mit dem öffentlichen Schlüssel zu verschlüsseln:

$$c \equiv m^e \pmod{N}$$

Die Entschlüsselung erfolgt in ähnlicher Weise mit dem privaten Schlüssel:

$$m \equiv c^d \pmod{N}$$

Dies führt aufgrund des Satzes von Euler-Fermat zu Erfolg:

$$x^{\varphi(N)} \equiv 1 \pmod{N}$$

Angewandt auf RSA:

$$m = c^d \equiv m^{e \cdot d} \pmod{N}$$
$$e \cdot d \equiv 1 \pmod{\varphi(N)}$$
$$m^{e \cdot d} \equiv m^{1 + x \cdot \varphi(N)} \pmod{N}$$
$$= m^1 \cdot m^{x \cdot \varphi(N)} \equiv m \cdot 1 \pmod{N}$$

Die Schwierigkeit von RSA besteht darin, vom öffentlichen Schlüssel auf den privaten Schlüssel zu schließen. Dies ist einfach, wenn N faktorisiert werden kann. Aktuell können Zahlen mit einer Länge um die 768 Bit faktorisiert werden. Für sicheres Arbeiten wird daher bei RSA eine Länge von mindestens 1024 oder besser 2048 Bit empfohlen (vgl. [44]).

Schwachstellen von RSA RSA hat bei einer Umsetzung, die dem Lehrbuch entspricht, eine Reihe von Schwachstellen (vgl. [36]):

Sollte $\varphi(N)$ bekannt werden, kann damit auf einfache Weise der private Exponent berechnet werden ($e \cdot d \equiv 1 \pmod{\varphi(N)}$). Außerdem ermöglicht dies die einfache Berechnung der Primfaktoren von N [36]. Ein häufig verwendeter kleiner öffentlicher Exponent e führt ebenso zu Problemen. Sollte eine Instanz anscheinend sinnlose Daten signieren, kann dies ausgenutzt werden. Dazu wird die sinnvolle Nachricht einfach mit einer beliebigen Zahl (r) hoch e multipliziert. Die Nachricht erscheint nun ohne sinnvollen Inhalt.

$$m' \equiv m \cdot r^e (mod(N))$$

$$s = m'^d \equiv r^{d \cdot e} \cdot m^d = r \cdot m^d \pmod{N}$$

Abschließend kann die Zahl r einfach entfernt werden. Weiters ist zu vermeiden, dass die zu verschlüsselnde Nachricht ein Vielfaches einer der beiden Primzahlen ist (vgl. [36]). Sollte dieser Zustand eintreten, ist das Ergebnis der Verschlüsselung wieder ein Vielfaches der Primzahl.Dazu ein kurzes Beispiel. Die Nachricht wird in diesem Fall als $p \cdot x$ angenommen. Wobei x eine beliebige positive ganzzahlige Zahl ist.

$$c \equiv (p \cdot x)^e \pmod{p \cdot q}$$
$$c + p \cdot q \cdot k = (p \cdot x)^e$$
$$c = (p \cdot x)^e - p \cdot q \cdot k$$
$$c = p \cdot (p^{e-1} \cdot x^e - q \cdot k)$$
$$X = p^{e-1} \cdot x^e - q \cdot k$$
$$c = p \cdot (X)$$

Weiters ist der RSA Algorithmus deterministisch. Sollten also die selbe Nachricht von einer Person ein zweites Mal verschlüsselt werden, führt dies zu zwei gleichen Ciphertexten. Aufgrund dieser Schwachstellen existieren zertifizierte Spezifikationen. Diese Spezifikationen sorgen außerdem dafür, dass Umsetzungen verschiedener Personen zu einander kompatibel sind (Padding usw.). Des Weiteren werden die Lehrbuchschwächen von RSA im Vorhinein ausgeschloßen. Die Spezifikation PKCS #1 v2.1 wird in Abschnitt 2.2.1.1 näher erläutert.

2.2.1.1 Praktische Umsetzung einer Verschlüsselung mit RSA

Sollen reale Daten mit RSA verschlüsselt werden, stellen sich, neben den oben genannten Problemen, einige Weitere. Wie soll zum Beispiel die Umwandlung von Bitfolgen in Zahlen zur Verschlüsselung erfolgen? Aus diesem Grund wurden einige praktisch orientierte Umsetzungen von RSA spezifiziert. Im Folgenden betrachten wir die Spezifikation PKCS #1 v2.1 [29], die von den RSA Laboratories erstellt wurde. Diese Spezifikation gliedert die Ver- und Entschlüsselung mit RSA in Teilbereiche.

OS2IP(X) und I2OSP($x, xLen$) Die Umwandlung von Bitfolgen in eine Zahl wird mit den Funktionen Integer-to-Octet-String (I2OSP, Umwandlung von Zahl in Bitfolge) und Octet-String-to-Integer(OS2IP, Umwandlung von Bitfolge in Zahl) behandelt. x repräsentiert die ganzzahlige, nicht negative Zahl in einer Darstellung zur Basis 256.

$$x = x_{xLen-1}256^{xLen-1} + x_{xLen-1}256^{xLen-1} + ... + x_{xLen-1}256^{xLen-1}$$

$xLen$ ist die geplante Länge der resultierenden Folge von Bytes. X stellt die resultierende Folge von Bytes dar. Zusätzlich muss noch überprüft werden ob $x \geq 256^{xLen}$ ist. Sollte dies der Fall sein, wird mit einer Fehlermeldung abgebrochen. Die Bitfolge (X) wird dabei als eine Anzahl von 8Bit langen Einheiten(Octets, Bytes) betrachtet.

$$X = X_1X_2X_3...X_{xLen}$$

Das Byte am linken Ende der Bitfolge wird dabei als das erste Byte bezeichnet und ist zugleich das höchstwertige. Das Byte am rechten Ende ist daher das niedrigstwertige und letzte Byte der Bitfolge. Wird nun eine nicht negative ganze Zahl in eine Bitfolge konvertiert, stellt x_{xLen-i} den Wert für das Byte X_i dar. Die Umwandlung einer Bitfolge in eine Zahl erfolgt in umgekehrter Weise (vgl. [29]).

RSAEP($(n, e), m$) und RSADP($(n, d), c$) Die Ver- und Entschlüsselung wird in zwei Subfunktionen realisiert (RSA Encryption Primitive, RSAEP und RSA Decription Primitive, RSADP). (n, e) stellen den öffentlichen RSA Schlüssel dar. Die Nachricht m und der Ciphertext c müssen kleiner als $n - 1$ und ein ganzzahliger positiver Integer sein. (n, d) stellen den privaten Schlüssel dar. Die Funktionen entsprechen hierbei im Großen und Ganzen der mathematischen Definition von RSA und sind im Folgenden dokumentiert (vgl. [29]):

$$c \equiv m^e \pmod{n}$$
$$m \equiv c^d \pmod{n}$$

Die genaue Ausführung der Berechnung ist dabei der jeweiligen Implementation überlassen.

RSAES-OAEP-ENCRYPT((n,e),M,L) Die bis jetzt behandelten Elemente sind die kleinsten in der Ver- und Entschlüsselung mit RSA. Zusätzlich wird noch ein Padding angewandt und alles in einer Funktion zusammengefasst. Diese Funktion wird als "RSA Encryption Scheme with Optimal Asymmetric Encryption Padding" (RSAES-OAEP) bezeichnet (vgl. [29]).

- **hLen**: Bytelänge des Hash-Werts der benutzten Hash-Funktion.
- **k**: Bytelänge des Modulus n bei der Verschlüsselung.
- **M**: Nachricht die verschlüsselt werden soll, mit einer maximalen Länge von $mlen \leq k - 2 \cdot hLen - 2$.
- **L**: Optionales Label für die Nachricht, Standardwert ist ein leerer String.
- **MGF**: Mask Generation Function

Zuerst wird überprüft ob L die maximale Inputläng der Hashfunktion überschreitet. Ist dies der Fall, wird mit einer Fehlermeldung abgebrochen. Sollte $mlen > k-2 \cdot hLen-2$ zutreffen, wird ebenfalls mit einer Fehlermeldung abgebrochen. Weiters wird aus L und der Hashfunktion der Wert $lHash$ errechnet:

$$lHash = Hash(L)$$

Anschließend wird ein Padding der Länge $k-mLen-2 \cdot hLen-2$ generiert. Dieses Padding besteht aus Bytes mit dem Wert 0. M, PS und $lHash$ werden zu einem Datenblock DB zusammengefasst:

$$DB = lHash \parallel PS \parallel 0x01 \parallel M$$

Weiters wird ein zufälliger Seed der Länge $hLen$ generiert. Mit diesem Seed und der Mask Generation Function (MGF) wird eine Maske $dbMask$ erstellt:

$$dbMask = MGF(seed, k - hlen - 1)$$

Diese Maske wird mit dem Datenblock per XOR verknüpft. Es entsteht der maskierte Datenblock $maskedDB$ ($maskedDB = DB \oplus dbMask$). Weiters wird eine Maske $seedMask$ für den Seed erstellt:

$$seedMask = MGF(maskedDB, hLen)$$

Diese Maske wird wiederum mit dem Seed per XOR verknüpft ($maskedSeed = Seed \oplus seedMask$). Zum Abschluss wird ein neuer Datenblock EM gebildet. Dieser besteht aus $maskedSeed$, $maskedDB$ und dem Byte $0x00$.

$$EM = 0x00 \parallel maskedSeed \parallel maskedDB$$

EM stellt die zu verschlüsselnde Nachricht dar. Die Bildung von EM wird in Abbildung 2.13 visualisiert (vgl. [29]).

Weiters wird EM nun einer Verschlüsselung zugeführt (vgl. [29]):

$$m = OS2IP(EM)$$
$$c = RSAEP((n, e), m)$$
$$C = I2OSP(c, k)$$

Das Endergebnis der Verschlüsselung c wird am Ende wieder in eine Folge von Bytes C umgewandelt. Die Entschlüsselung erfolgt analog:

RSAES-OAEP-DECRYPT((n,d),C,L) Die Funktion RSAES-OAEP-DECRYPT ist für die Entschlüsselung zuständig. Daher wird ähnlich wie bei der vorigen Funktion vorgegangen. Zuerst wird überprüft, ob L die maximale Inputgröße überschreitet, C eine Länge ungleich k Bytes aufweißt oder $k < 2hlen + 2$ zutrifft. Beim Zutreffen einer oder mehrerer dieser Fälle wird mit einer Fehlermeldung abgebrochen. Anschließend wird C in eine Zahl umgewandelt, entschlüsselt und das Ergebnis wieder in eine Folge von Bytes konvertiert (vgl. [29]):

$$c = OS2IP(C)$$
$$m = RSADP((n, d), c)$$

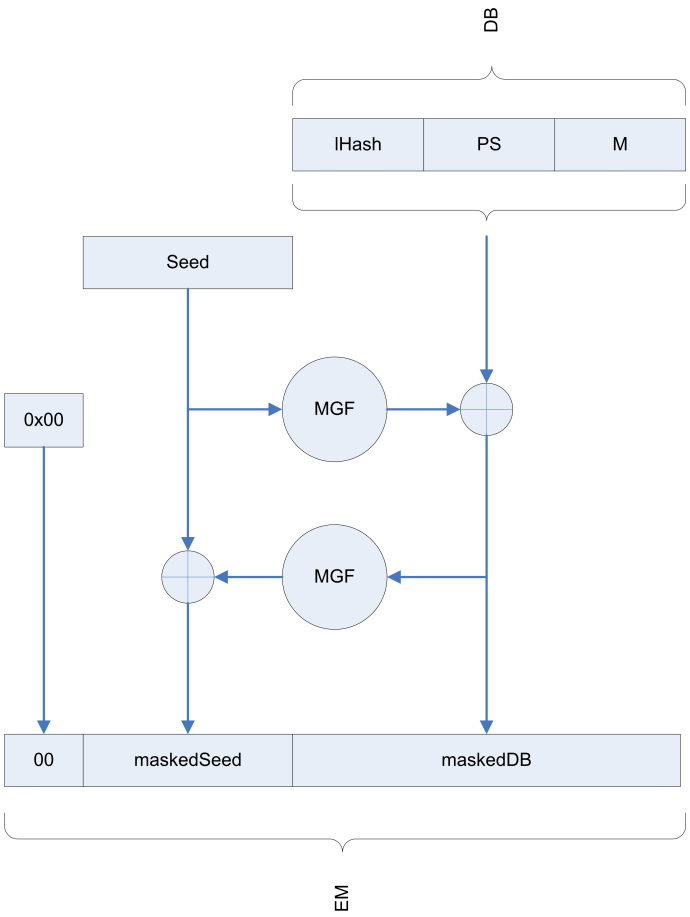

Abbildung 2.13: Bildung der Encapsulated Message (EM) für RSAES-OAEP [29]

$$EM = I2OSP(m, k)$$

Aus L wird nun analog zur Verschlüsselung $lHash$ gebildet. Danach wird der Datenblock EM wieder in seine Bestandteile aufgeteilt:

$$EM = Y \parallel maskedSeed \parallel maskedDB$$

Sollte Y nicht dem Byte $0x00$ entsprechen, wird mit einer Fehlermeldung abgebrochen. Weiters wird aus $maskedDB$ mittels der Mask Generation Function $seedMask$ gewonnen:

$$seedMask = MGF(maskedDB, hLen)$$

Die Verknüpfung von $maskedSeed$ und $seedMask$ mittels XOR gibt den $Seed$ preis ($seed = maskedSeed \oplus seedMask$). Aus dem nun verfügbaren Seed wird in weiterer Folge analog zur Verschlüsselung $dbMask$ erzeugt. $dbMask$ wird nun mit $maskedDB$ per XOR verknüpft und damit wird das Datenpaket DB erzeugt. Abschließend wird DB wieder in seine Bestandteile aufgeteilt ($DB = lHash' \parallel PS \parallel 0x01 \parallel M$). Es folgt die Überprüfung von $lHash'$ ($lHash'$ muss $lHash$ entsprechen). Sollte dies nicht der Fall sein wird mit einer Fehlermeldung abgebrochen. Abgebrochen wird zusätzlich, wenn das Byte $0x01$ in DB nicht vorhanden ist. Tritt keiner dieser Fehler auf, war die Entschlüsselung erfolgreich und die Nachricht M wird zurückgegeben (vgl. [29]).

2.2.1.2 Analyse des RSA

Bei Beachtung der möglichen Angriffspunkte und der Wahl eines ausreichend großen Modulus N, stellt RSA eine sichere asymmetrische Möglichkeit der Verschlüsselung dar. Großer Wert ist dabei auf eine spezifizierte Implementation zu legen. Dies minimiert die Wahrscheinlichkeit für Fehler und macht die Umsetzung mit anderen Systemen interoperabel. Das Einsatzgebiet von RSA erstreckt sich aufgrund des Rechenaufwands vor allem auf den Schlüsselaustausch und das Signieren von Nachrichten.

2.2.2 Asymmetrische Kryptographie mit dem diskreten Logarithmus

Als Alternative zu RSA hat sich in den letzten Jahren Kryptographie basierend auf der Berechung des diskreten Logarithmus etabliert. Dazu gibt es einige Umsetzungen die auf verwandten Problemen basieren. Ein Beispiel dafür ist der ElGamal Algorithmus. Im Folgenden wird dieser Algorithmus kurz erläutert.

Am Anfang wird eine große Primzahl p (um die 1024 Bit) benötigt. $p-1$ muss durch eine weitere möglichst große Primzahl (um die 160 Bit) teilbar sein. Weiters wird ein Generator g (näheres dazu unter [44]) gewählt. Als privater Schlüssel wird eine beliebige ganzzahlige Zahl x kleiner als $p-2$ gewählt. Der öffentliche Schlüssel h errechnet sich wie folgt:

$$A \equiv g^x \pmod{p}$$

Der gesamte öffentliche Schlüssel besteht nun aus der Primzahl p, dem Generator g und dem errechneten A. Zur Verschlüsselung wird nun ein zufälliger einmaliger Schlüssel k erzeugt.

$$c_1 \equiv g^k \pmod{p}$$
$$c_2 \equiv m \cdot A^k \pmod{p}$$

Der Ciphertext besteht nun aus (c_1, c_2). Für die Verschlüsselung wird der private Schlüssel x benötigt:

$$m \equiv \frac{c_2}{c_1^x} \pmod{p}$$
$$m \equiv \frac{m \cdot A^k}{g^{k \cdot x}} \equiv \frac{m \cdot g^{x \cdot k}}{g^{k \cdot x}} \equiv m \pmod{p}$$

Die Sicherheit bei El Gamal definiert sich durch die Schwierigkeit von A auf den privaten Schlüssel x zu schließen. Die Lehrbuchvariante von El Gamal ist jedoch nicht gegen alle Attacken sicher. Daher wird in der Praxis eine modifizierte Version verwendet (vgl. [44]).

Ein weiteres Beispiel ist die "Elliptic Curve Cryptography" (ECC). Bei ECC beruht die Sicherheit auf der Berechnung des diskreten Logarithmus auf elliptischen Kurven. ECC hat in den letzten Jahren stark an Bedeutung gewonnen. Im Vergleich zur Primzahlenzerlegung gilt das Lösen des diskreten Logarithmus auf elliptischen Kurven als ungleich schwerer. Daher kann eine ähnliche Sicherheit bei wesentlich kürzeren Schlüsseln erreicht werden (Im Vergleich der benötigten Rechendauer um das dahinterliegende mathematische Problem zu lösen, entsprechen 160Bit ECC einer Schlüsselänge von 1024Bit bei RSA) (vgl. [46]). ECC gilt zudem als weniger rechenaufwändig als RSA und ist daher eine Alternative beim Einsatz auf Systemen mit geringen Ressourcen. Jedoch hat ECC erst an Beachtung gewonnen und ist daher im Gegensatz zu RSA weniger gut untersucht (vgl. [37]).

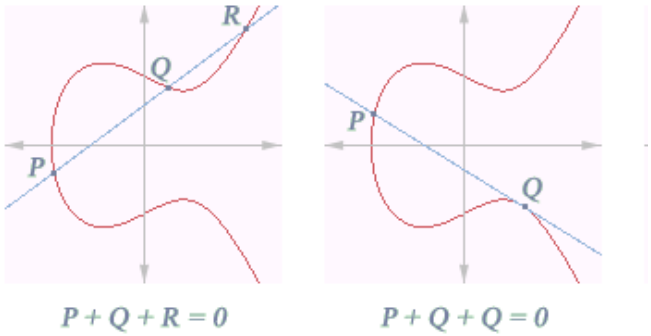

Abbildung 2.14: Addition auf Elliptischen Kurven [15]

2.3 Hash-Funktionen

Eine Hash-Funktion generiert aus einer Nachricht beliebig langer Daten einen Fingerabdruck (Hash-Wert) begrenzter Länge. Dieser Fingerabdruck wird genutzt um Änderungen an den Daten sichtbar zu machen (Eine kleine Änderung der Daten führt zu einem völlig anderem Fingerabdruck, siehe Abbildung 2.15). Daher werden Hash-Funktionen auch als Modifikation Detection Codes (MDC) bezeichnet. Eine andere Art der Verwendung ist die Einwegverschlüsselung von Daten mit dem Ziel, dass aus dem Hash-Wert nicht mehr auf die dahinter liegenden Daten rückgeschlossen werden kann. Um aus kryptologischer Sicht sicher zu sein, muss eine Hash-Funktion eine Reihe von Eigenschaften erfüllen (vgl. [46]).

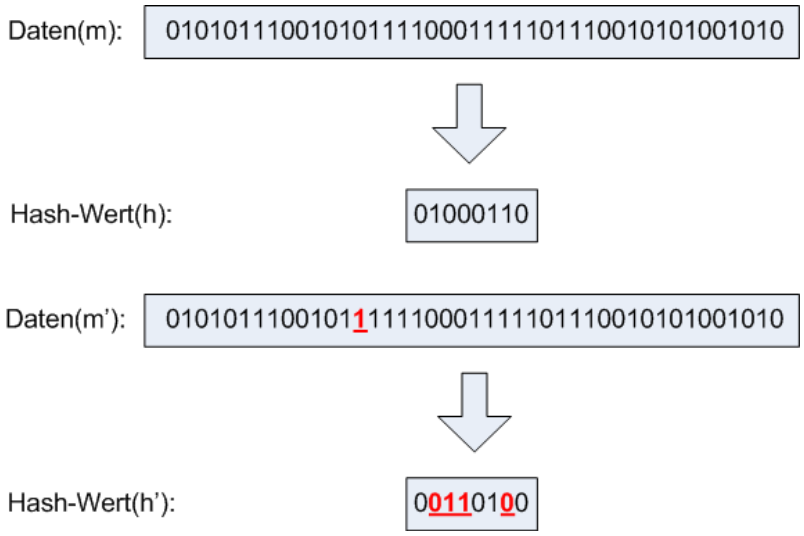

Abbildung 2.15: Hash-Funktionen: Änderungen bei den Daten führen zu Änderungen im Hash-Wert.

Die Hashfunktion soll unabhängig von der Länge der Daten anwendbar sein. Das heißt sie muss Daten beliebiger Länge akzeptieren und daraus einen Hash-Wert errechnen. Im Gegensatz muss der erzeugte Hash-Wert unabhängig von der Länge der Daten immer die gleiche Länge aufweisen. Dieser Hash-Wert muss auf solche Weise generiert werden, dass ein Rückschluss auf die Daten durch den Hash-Wert nicht möglich ist (one-way function). Des Weiteren soll es für eine gegebene Nachricht unmöglich sein, eine zweite unterschied-

liche Nachricht zu finden, die zum selben Hash-Wert führt (weak collision resistance). Außerdem sollte es unmöglich sein zwei Nachrichten zu finden, die den selben Hash-Wert erzeugen (strong collision resistance) (vgl. [46]).

Es existieren einige Algorithmen, die diese Forderungen umsetzen. Eigens für die Berechnung von Hash-Werten wurden die Algorithmen der SHA Familie entwickelt. Im folgenden Abschnitt werden zwei Vertreter dieser Familie, SHA-1 und SHA-256, detailliert erklärt.

2.3.1 SHA-1

SHA (Secure Hash Algorithm) wurde im Jahre 1993 vom National Institut of Standards and Technology entwickelt und als Standard (FIPS 180) festgelegt (vgl. [33]). Nach einer Überarbeitung 1995 wird der Algorithmus als SHA-1 bezeichnet. SHA-1 generiert aus einer beliebig langen Bitfolge (maximal jedoch eine Länge von 2^{64} Bits) einen 160 Bit langen Hash-Wert. Die Blöcke werden auf ein Vielfaches der Blocklänge gepaddet. Das Padding besteht aus einem einzelnen Bit mit dem Wert 1 und einer Anzahl von Bits mit dem Wert 0, bis die notwendige Länge erreicht ist. Abschließend wird der Wert der Nachrichtenlänge mit 64 Bit angehängt. Das Padding besteht aus einem einzelnen Bit mit dem Wert 1 und einer Anzahl von Bits mit dem Wert 0 bis die notwendige Länge erreicht ist. Abschließend wird ein Wert der Länge 64 Bit angehängt. Dieser Wert repräsentiert die ursprüngliche Länge der Bitfolge vor dem Padding. Der Block wird nun in 16 Wörter der Länge 32 Bit (W_0 bis W_{15}) aufgeteilt. Aus diesen Wörtern werden weitere 64 Wörter (W_{16} bis W_{79}) mittels folgender Formel generiert (Für $16 \leq t \leq 79$):

$$W_t = <<<^1 (W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$$

Weiters wird ein 160 Bit großer Buffer bestehend aus 5 Wörtern (A, B, C, D, E) angelegt und mit den Startwerten h_1 bis h_0 initialisiert.

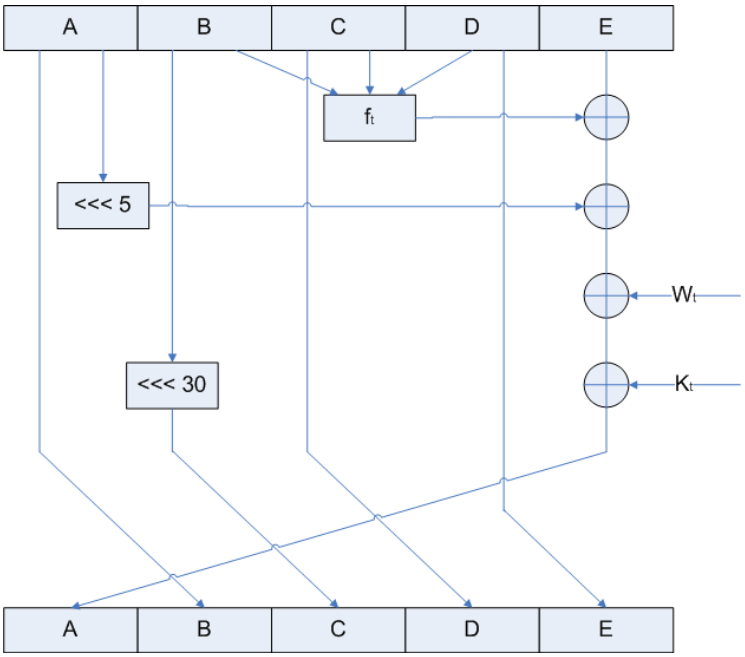

Abbildung 2.16: Kompressionsrunde bei SHA-1 [33]

Daraus wird in weiterer Folge in insgesamt 80 Runden ($0 \leq t \leq 79$) der Wert des Buffers anhand folgender Funktion errechnet:

$$\begin{aligned} A =& f_t(B,C,D) \oplus <<<^5 (A) \oplus W_t \oplus K_t \\ B =& A \\ C =& <<<^{30} (B) \\ D =& C \\ E =& D \end{aligned}$$

Der Ablauf einer Runde wird in Abbildung 2.16 illustriert. Die Konstante K ist abhängig von der Runde t (näheres dazu unter [33]). Die Funktion $f(B,C,D)$ ist ebenso abhängig von der Runde t und ist wie folgt definiert:

$$\begin{aligned} 0 \leq t \leq 19 \quad & f(B,C,D) = (B \wedge C) \vee (\neg(B) \wedge D) \\ 20 \leq t \leq 39 \quad & f(B,C,D) = B \oplus C \oplus D \\ 40 \leq t \leq 59 \quad & f(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \vee \\ 60 \leq t \leq 79 \quad & f(B,C,D) = B \oplus C \oplus D \end{aligned}$$

Nach Verarbeitung der 80 Runden wird abschließend der Buffer (A,B,C,D,E) nochmals mit dem Ergebnis des vorigen Blocks addiert (siehe Abbildung 2.17). Nach Abschluss aller Blöcke stellt A,B,C,D,E den 160 Bit Hash-Wert dar (vgl. [33]).

Angriffe und Sicherheit Die Sicherheit von SHA-1 hängt von der Möglichkeit Kollisionen zu finden ab. So wurde eine Methode, die bei SHA-0 zu Erfolg führte, 2005 von Wang et al. auf SHA-1 angewandt (vgl. [51]). Weitere Untersuchungen einer anderen Gruppe führten zu einem theoretischen Angriff auf eine reduzierte Version von SHA-1 mit 53 Runden (vgl. [39]). Diese Angriffe wurden in den folgenden Jahren verfeinert (vgl. [12, 11]). 2008 präsentierten Rechberger und De Cannière eine Methode, bei der 25% des Textes frei wählbar sind. Im Gegensatz zu vorigen Angriffen, wo das Ergebnis aus einem Paar von unleserlichen Datensätzen bestand, ist dies ein Meilenstein in Richtung praktische Angriffe auf SHA-1. Jedoch wurde auch in diesem Fall nur mit einer reduzierten Version von SHA-1 gearbeitet. Laut Autor ist die gleiche Methode jedoch auch auf SHA-1 mit 80 Runden anwendbar (vgl. [10]).

2.3.2 SHA-256

Im Jahre 2001 wurden weitere Algorithmen der SHA-Familie veröffentlicht und werden als SHA-2 bezeichnet (vgl. [33]). Dazu gehören SHA-224, SHA-256, SHA-384 und SHA-512. Diese Algorithmen bauen auf die selbe Struktur auf und unterscheiden sich nur durch die Anzahl der Runden, benutzten Verknüpfungen, verwendeten Konstanten und eine nichtlineare Message Expansion. In weiterer Folge wird die Struktur und Funktion der SHA-2 Gruppe anhand von SHA-256 erläutert. SHA-256 arbeitet mit einer Blockgröße von 512 Bit. Im Gegensatz zu SHA-1 arbeitet SHA-256 jedoch mit einer internen Buffergröße von

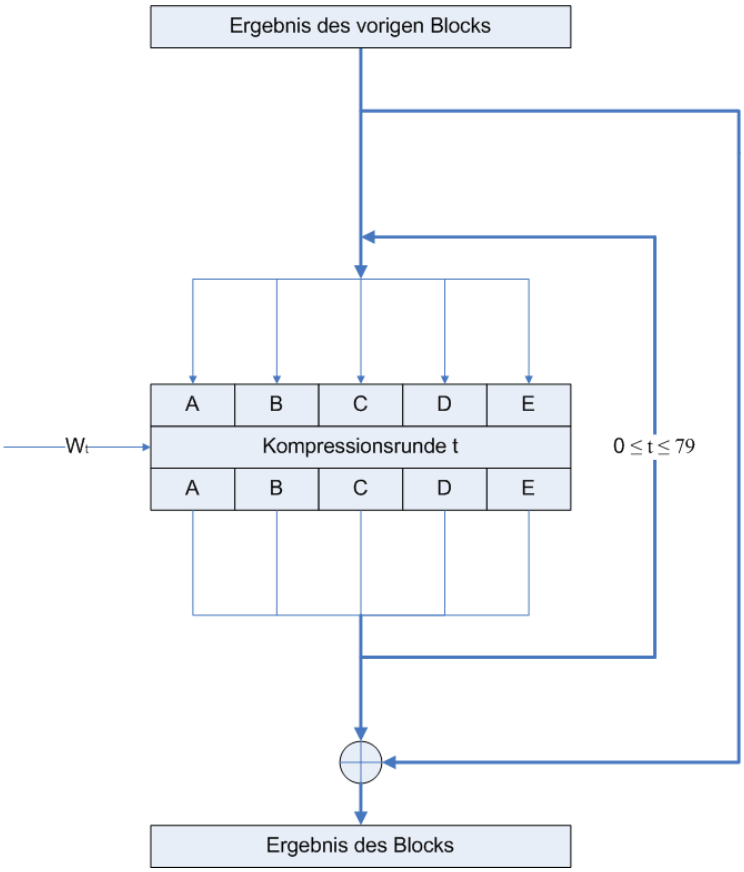

Abbildung 2.17: Kompression eines Datenblocks bei SHA-1

256 Bit, was schlussendlich auch die Größe des finalen Hash-Werts ist. Die maximale Nachrichtenlänge beträgt wie bei SHA-1 2^{64} Bit. Das Padding erfolgt analog zu SHA-1. Der enstandene Block wird in 16 Wörter der Länge 32 Bit (W_0 bis W_{15}) aufgeteilt. Aus diesen Wörtern werden weitere 48 Wörter (W_{16} bis W_{63}) mittels folgender Formel generiert (Für $16 \leq t \leq 63$)(vgl. [33]):

$$W_t = \sigma_1^{\{256\}}(W_{t-2}) \oplus W_{t-7} \oplus \sigma_0^{\{256\}}(W_{t-15}) \oplus W_{t-16})$$

Die Funktionen $\sigma_0^{\{256\}}$, $\sigma_1^{\{256\}}$, $\sum_0^{\{256\}}$ und $\sum_1^{\{256\}}$, $Ch(x, y, z)$, $Maj(x, y, z)$ verknüpfen die einzelnen Bufferwerte.

Weiters fließen eine Reihe von Konstanten ($K_0^{\{256\}} - K_{63}^{\{256\}}$) in die Kompressionsrunden ein. Damit wird in weiterer Folge in insgesamt 64 Runden ($0 \leq t \leq 63$) der Wert des Buffers anhand einer Anzahl von Funktionen berechnet. Die Funktionen sind im Standard [33] definiert.

Abschließend nach Verarbeitung der 80 Runden wird der Buffer (A, B, C, D, E, F, G, H) nochmals mit dem Ergebnis des vorigen Blocks addiert. Nach Abschluß aller Blöcke stellt A, B, C, D, E, F, G, H den 160 Bit Hash-Wert dar.

Angriffe und Sicherheit Da für SHA-1 mittlerweile eine Reihe von Angriffen bekannt sind, bietet sich SHA-256 als zur Zeit sichere Alternative an. Trotzdem sind bereits Attacken mit einer reduzierten Anzahl von Runden in Arbeit (vgl. [42]). NIST hat mittler-

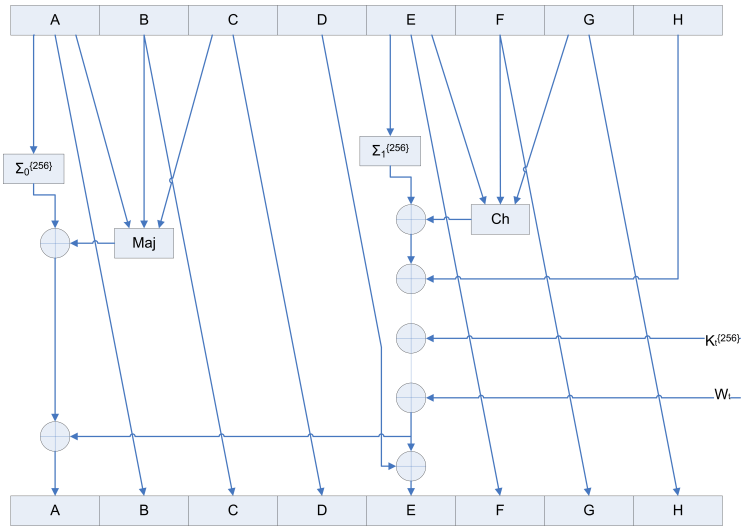

Abbildung 2.18: Kompressionsrunde bei SHA-256 [33]

weile den Wettbewerb für SHA-3 gestartet, da sich SHA-1 und SHA-2 zu ähnlich sind. Bisher sind jedoch keine direkten Erweiterungen von SHA-1 Attacken auf SHA-2 bekannt.

2.3.3 HMAC

Bezieht man nun einen Schlüssel mit in die Generierung des Hash-Werts ein, ist zusätzlich eine Authentifizierung inkludiert. Dies wird als Message Authentication Code (MAC) bezeichnet. Ein Beispiel für einen solchen MAC ist HMAC. Dabei wird eine beliebige Hash-Funkion benutzt um einen Message Authentication Code zu erstellen. HMAC liegt als Standard RFC 2104 vor. Mathematisch ausgedrückt hat HMAC folgende Form (vlg. [46]):

$$HMAC(k, m) = H[(k^+ \oplus opad) \| H[(k^+ \oplus opad) \| M]]$$

- k bezeichnet in diesem Fall den Schlüssel und k^+ den auf Blocklänge gepaddeten Schlüssel.
- m bezeichnet die Nachricht.
- *opad* bezeichnet eine gewisse Bitfolge (01011010).
- *ipad* bezeichnet eine gewisse Bitfolge (00110110).
- H bezeichnet die verwendete Hash-Funktion.

Zuerst wird der Schlüssel k mit 0 auf Blocklänge gepaddet. Dieser Schlüssel wird einmal mit der Bitfolge *opad* verknüpft. Weiters wird der Schlüssel mit *ipad* verknüpft und der Nachricht vorangestellt gehashed. Diesem Hash Wert wird nun der mit *opad* gepaddete Schlüssel vorangestellt und die ganze Konstruktion nochmals gehashed. Dies ergibt den Message Authentication Code (siehe Abildung 2.19) (vlg. [46]).

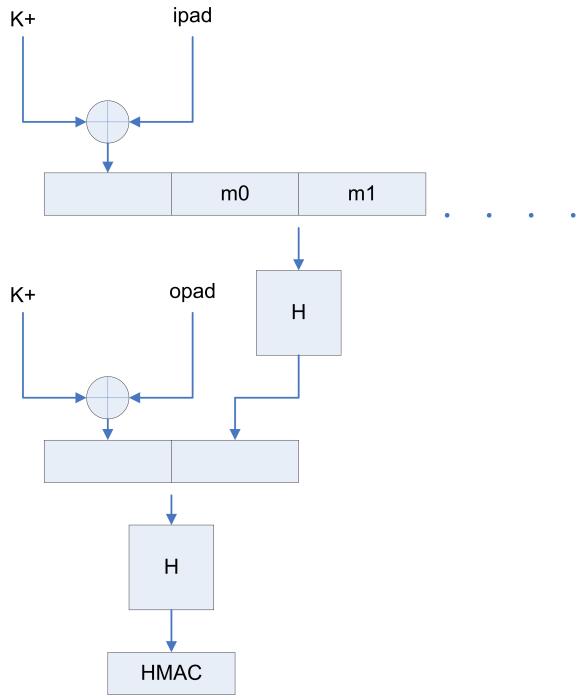

Abbildung 2.19: Konstruktion eines MAC mittels HMAC [46]

2.4 Digitale Signaturen

Um den Ursprung einer Nachricht prüfbar zu machen, wurde das Prinzip der digitalen Signaturen entwickelt. So kann mit Hilfe asymmetrischer Kryptographie durch den Sender bewiesen werden, dass eine Nachricht von ihm stammt. Dies löst das am Anfang dieses Kapitels erwähnte Problem der Authentifizierung von Kommunikationspartnern. Richtig implementierte digitale Signaturen stellen somit das digitale Äquivalent zur Unterschrift dar. Um dies zu ermöglichen, muss eine digitale Signatur eine Reihe von Eigenschaften aufweisen (vlg. [46]).

Autor, Datum und Zeit einer Signatur müssen überprüfbar sein. Dasselbe gilt für den signierten Inhalt. Auch dieser muss durch die Signatur verifizierbar sein. Außerdem muß eine Signatur zusätzlich von Dritten überprüfbar sein. Dies hat den Zweck, dass im Falle eines Streits eine dritte Instanz klärend einschreiten kann. Die Erstellung und Überprüfung der Signatur muss zudem auf einfache Weise und für jedermann möglich sein. Dabei muß jedoch die Möglichkeit zur Fälschung einer digitalen Signatur ausgeschlossen sein (vlg. [46]).

Weiters unterscheidet man zwei Formen von digitalen Signaturen:

- Signaturen mit Message Recovery
- Signaturen mit Appendix

Bei digitalen **Signaturen mit Message Recovery** kann die Nachricht aus der Signatur wiederhergestellt werden. Die Nachricht muss nicht neben der Signatur übertragen werden (siehe Abbildung 2.20).

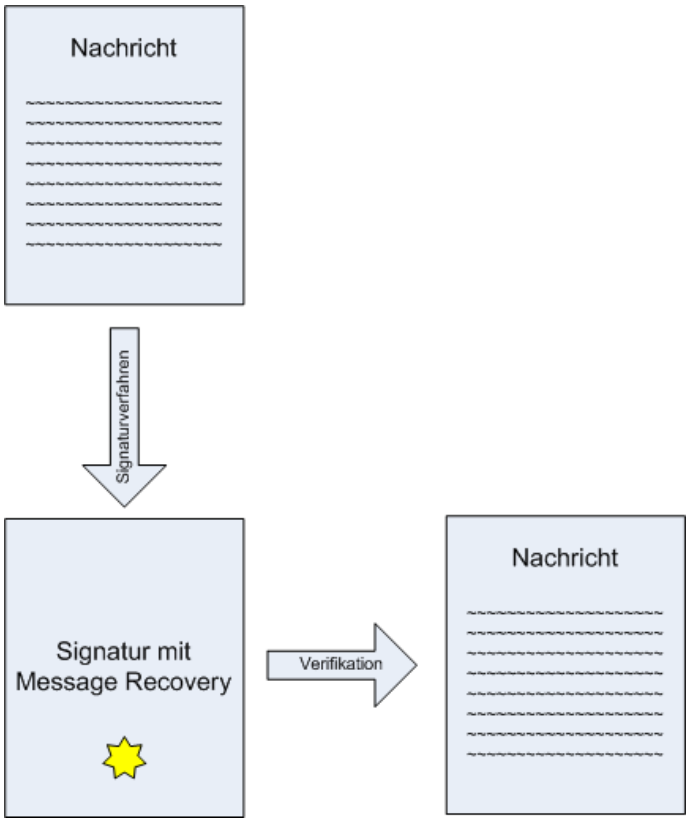

Abbildung 2.20: Signatur mit Message Recovery

Bei **Signaturen mit Appendix** wird eine Zusammenfassung (ein Hash-Wert der Nachricht) signiert. Um die Signatur verifizieren zu können, muss die Nachricht mitübertragen werden (siehe Abbildung 2.21).

Wie bereits in Abschnitt 2.2 beschrieben, existiert für jeden Kommunikationsteilnehmer ein Schlüsselpaar, bestehend aus öffentlichem und privatem Schlüssel. Mit Hilfe asymmetrischer Kryptographie wird nun für eine Nachricht eine Signatur erstellt. Die Nachricht sollte Datum, Zeit und Signator enthalten, um die oben genannten Punkte zu erfüllen. Wird zur Signierung eine Hash-Funktion verwendet, muss es unmöglich sein, eine Nachricht zu finden die den selben Hash-Wert wie die signierte Nachricht aufweist (mehr dazu in Abschnitt 2.3).

Des Weiteren wird auf die Nachricht m ein Padding angewandt. Dies bedeutet, dass die Nachricht auf eine festgelegte Länge mit einem bestimmten Bitmuster aufgefüllt wird. Die enstandene Nachricht m' wird nun mit dem privaten Schlüssel verschlüsselt oder es wird daraus ein Hash-Wert h errechnet und dieser im weiteren Verlauf verschlüsselt. Bei digitalen Signaturen mit Message Recovery muss zur Verifikation die erhaltene Signatur s mit dem öffentlichen Schlüssel entpackt und der Inhalt überprüft werden. Bei Signaturen mit Appendix muss die Nachricht gepaddet und mittels Hash-Funktion ein Hash-Wert errechnet werden. Die erhaltene Signatur wird mit dem öffentlichen Schlüssel entpackt und mit dem errechneten Hash-Wert verglichen (vlg. [46]).

Digitale Signaturen mit Message Recovery haben einige Nachteile. So ist bei langen Nachrichten die Verschlüsselung und Entschlüsselung mit asymmetrischer Kryptographie sehr aufwändig. Des Weiteren müsste zur Archivierung die gesamte Nachricht und die

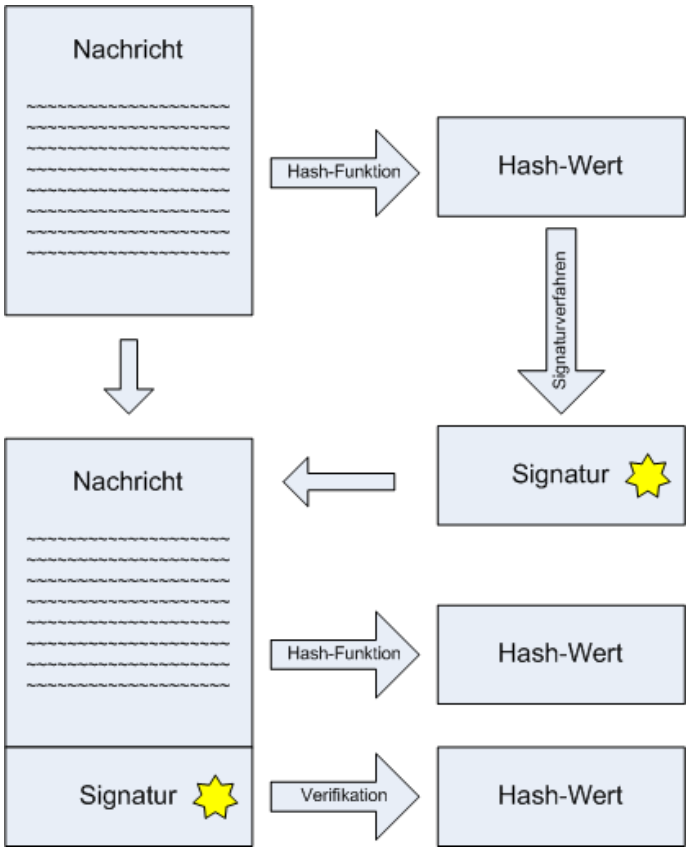

Abbildung 2.21: Signatur mit Appendix

Signatur (gleich lang wie die Nachricht) gespeichert werden (vlg. [46]). Daher wird bei den nächsten Beispielen mit Signaturen mit Appendix gearbeitet. Als Hash-Funktion kommt SHA-1 zum Einsatz. Mehr zur Hash-Funktion und zum angewandten Padding in Abschnitt 2.3.1.

2.4.1 Signaturen mit RSA

Um eine Signatur mit RSA zu erstellen, werden wie bei normaler Verschlüsselung zwei große Primzahlen p und q gewählt. Diese bilden den öffentlichen Modulus $n = p \cdot q$. Weiters wird ein öffentlicher Exponent e ($gcd(e, \varphi(n) = 1)$ gewählt und der private Exponent d berechnet. Um eine Signatur s zu einer Nachricht m zu erstellen, wird nun aus dieser zuerst ein Hash-Wert h errechnet:

$$m' = PAD(m)$$
$$h = HASH(m')$$

Weiters wird dieser Hash-Wert mit dem privaten Schlüssel signiert:

$$s \equiv h^d \pmod{N}$$

Zum Verifizieren wird nun wieder die Signatur mit dem öffentlichen Schlüssel verifiziert:

$$h' \equiv s^e \equiv h^{e \cdot d} \pmod{n}$$

Der Empfänger muss jetzt nur noch den Hash-Wert aus der Nachricht generieren und diesen mit der Signatur vergleichen ($h' = h$). Gut ersichtlich ist, dass RSA in diesem Fall auch für die Verschlüsselung der Nachricht verwendet werden kann [44].

Für eine praxisgerechte Umsetzung von Signaturen mit RSA bedarf es jedoch einiger zusätzlicher Maßnahmen. Diese sind in der Spezifikation PKCS #1 v2.1 [29] und im "Digital Signature Standard" (DSS) [34] festgehalten. In PKSC #1 wird die genaue Vorgehensweise zur Erstellung von Signaturen mit RSA definiert. Diese Vorgehensweise soll im folgenden kurz skizziert werden. In PKSC sind Elemente definiert, die zusammen den Algorithmus ergeben. Die Module OS2IP und I2OSP wurden bereits in Abschnitt 2.2.1.1 näher definiert. Weiters sind noch Module zum Signieren notwendig. Diese werden als RSASP1 (RSA Signature Primitive) und RSAVP1 (RSA Verification Primitive) bezeichnet (vgl. [29]):

- **(n,e)**: Der öffentliche RSA Schlüssel.
- **(n,d)**: Der private RSA Schlüssel.
- **m**: Die Nachricht in der Form eines ganzzahligen nicht negativen Integers, der kleiner als $n - 1$ sein muss.
- **s**: Signatur, ganzzahliger nicht negativer Integer, ebenfalls kleiner als $n - 1$.
- **M**: Nachricht, als Bitfolge.
- **S**: Signatur, als Bitfolge.
- **k,modBits**: Bitlänge des Modulus n.

Zur Signierung selbst und zur Überprüfung der Signatur sind zwei Funktionen definiert:

RSASP1((n,d),m)

$$s \equiv m^d \pmod{n}$$

RSAVP1((n,e),s)

$$m \equiv s^e \pmod{n}$$

Beide Funktionen sind equivalent zur Lehrbuch-Signatur mit RSA definiert. Die technische Umsetzung wird dabei der Implementation überlassen. Der Vorteil gegenüber RSA laut Lehrbuch, liegt in der Art und Weise wie die Daten verpackt werden. Dieses Verpacken der Nachricht wird mittels EMSA-PSS (Encoding Method for Signatures with Appendix - Probabilistic Signature Scheme)durchgeführt (siehe Abbildung 2.22) (vgl. [29]):

Zu aller erst wird überprüft ob M die maximale Inputgröße der Hashfunktion überschreitet. Ist dies der Fall, ist ein Signieren nicht möglich und der Algorithmus bricht mit einer Fehlermeldung ab. Aus der Nachricht M wird in weiterer Folge mittels einer Hash-Funktion ein Hash-Wert gebildet ($mHash = Hash(M)$). Zusätzlich wird ein *salt* der Länge *sLen* generiert. Wurde *sLen* mit 0 angegeben, so ist *salt* der leere String. Trifft dies nicht zu, ist *salt* eine zufällige Bitfolge mit der definierten Länge. Aus *salt*, $mHash$ und einer Anzahl von Bytes mit dem Wert $0x00$ wird ein Paket M' erstellt:

$$M' = (0x)00 \quad 00 \quad 00 \quad 00 \quad 00 \quad 00 \quad 00 \quad 00 \parallel mHash \parallel salt$$

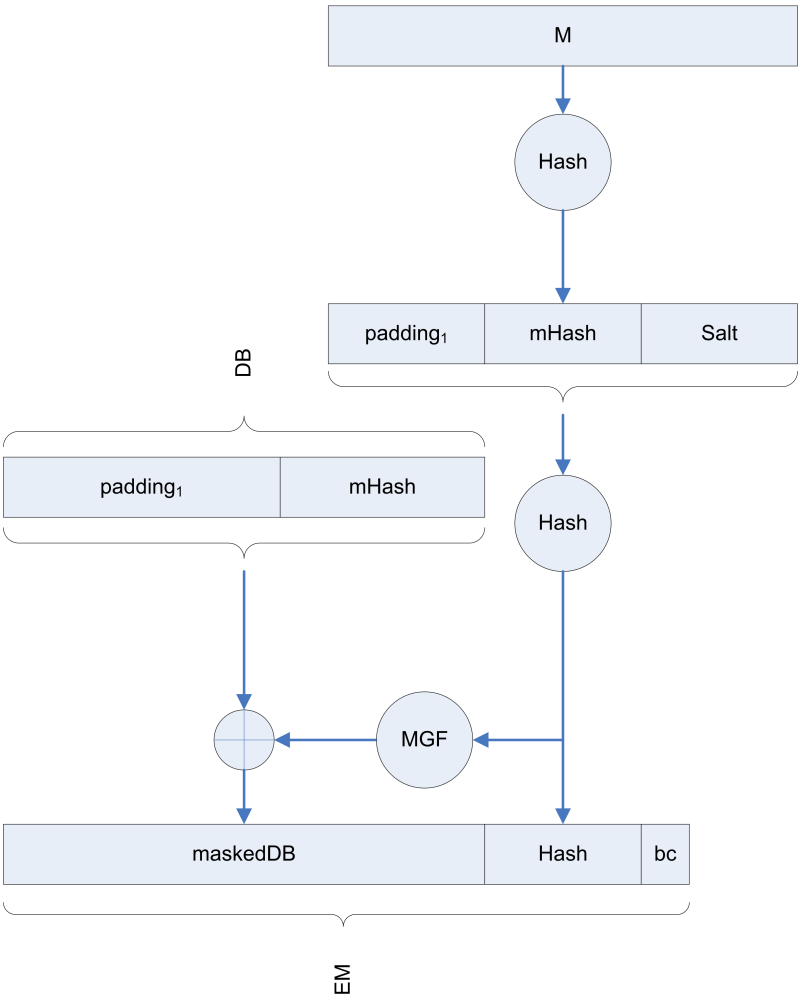

Abbildung 2.22: Encoding der Nachricht mittels EMSA-PSS [29]

Aus M' wird nun nochmals ein Hash-Wert H errechnet ($H = Hash(M')$). Weiters wird nun ein String PS der Länge $emLen - sLen - hLen - 2$ generiert. Die Bytes des Strings PS haben den Wert $0x00$. Aus PS und $salt$ wird nun DB gebildet ($DB = PS \parallel 0x01 \parallel salt$). Zusätzlich muss noch eine Maske für DB generiert werden ($dbMask$):

$$dbMask = MGF(H, emLen - hLen - 1)$$

Die Maske und DB werden nun noch per XOR verknüft ($maskedDB = DB \oplus dbMask$). Abschließend werden $8 \cdot emLen - emBits$ der höchstwertigen Bits von $maskedDB$ auf 0 gesetzt und mit H und dem Byte $0xbc$ EM gebildet (vgl. [29]):

$$EM = maskedDB \parallel H \parallel 0xbc$$

Die nun definierten Funktionen werden in weitere Folge benutzt um eine Signatur zu erstellen oder zu überüfen. Dazu sind folgende zwei Funktionen definiert:

RSASSA-PSS-SIGN ((n,d), M) Auf die Nachricht M wird zuerst das in Abbildung 2.22 dargestellte Encoding-Schema angewandt ($EM = EMSA-PSS(M, emBits)$). Weiters wird EM in eine Zahl umgewandelt, die Signierung durchgeführt und abschließend

wieder in eine Byte-String umgewandelt:

$$m = OS2IP(EM)$$

$$s = RSASP1((n, d), m)$$

$$S = I2OSP(s, k)$$

Die Funktion RSASSA-PSS-VERIFY dient zur Verifikation der zuvor erstellen Signatur:

RSASSA-PSS-VERIFY ((n, e), M, S) Anfangs wird überprüft, ob die Länge der Signatur S der Länge k des Modulus n entspricht. Ist dies nicht der Fall wird eine Fehlermeldung zurückgegeben. Anschließend wird die Signatur in eine Zahl umgewandelt und diese wieder in eine Representation der Nachricht m umgewandelt.

$$s = OS2IP(S)$$

$$m = RSAVP1((n, e), s)$$

Diese Representation wird nun wieder in eine Darstellung in Form von Bytes umgewandelt.

$$EM = I2OSP(m, emLen)$$

Mittels EMSA-PSS-Verification wird überprüft, ob EM der Nachricht entspricht.

$$Ergebnis = EMSA - PSS - VERIFY(M, EM, modBits - 1)$$

Die Funktion EMSA-PSS-VERIFY entpackt EM in umgekehrter Reihenfolge wie die Funktion EMSA-PSS und vergleicht das Ergebnis mit dem Hash-Wert der Nachricht.

Zusätzlich zu dieser Spezifikation in PKSC #1 sind im Digital Signature Standard (DSS) weitere Anforderungen für eine sichere Signatur definiert. Zum Beispiel darf ein Schlüsselpaar nur für eine Signatur-Methode verwendet werden. Eine Verwendung für verschiedene Zwecke würde die Sicherheit des Schlüsselpaars gefährden. Der Modulus n ist in diesem Schema nur aus zwei Primzahlen zusammengesetzt (p und q). Eine Mulitplikation von mehreren Primzahlen ist unzulässig. Des Weiteren ist die Länge des Modulus auf 1024, 2048 oder 3072 festgelegt. Die im Schema verwendete Hash-Funktion muss im Secure HashStandard ([33]) spezifiziert sein. Eine ähnliche Anforderung gilt an den Schlüsselgenerator. Dieser muss von anerkannter Stelle zertifiziert sein. Außerdem soll die Länge des Salts ($sLen$) bei RSASSA-PSS $0 \leq sLen \leq hLen$ betragen (vgl. [34]):

2.5 Protokolle

Rund um den kryptographischen Algorithmus gibt es unzählige Punkte, welche die Sicherheit eines Systems bestimmen. Zum einen ist die Implementierung des Algorithmus von großer Bedeutung, da bestimmte Attacken auf Fehler in dieser aufbauen. Zum Anderen macht ein Algorithmus alleine noch kein System. Für den richtigen Einsatz gibt es Anleitungen die angeben, wie und in welchem Umfeld der Algorithmus eingesetzt werden soll. Dies wird in der Kryptographie als Protokoll bezeichnet.

Im folgenden Abschnitt werden Protokolle zur symmetrischen Verschlüsselung von Datenströmen, zur Verwaltung von Schlüsselpaaren, sowie zur Authentifizierung und Verschlüsselung mittels asymmetrischer Kryptographie vorgestellt. Zu allererst müssen jedoch

die möglichen Angriffe definiert werden. Sieht man den Algorithmus selbst als perfekten Algorithmus ohne Schwächen, der auch in solcher Form implementiert wurde, hat ein Angreifer noch immer einige Möglichkeiten:

- Replay Attacke
- Man in the Middle Attacke
- Reflection Attacke

Replay Attacke: Daten einer aufgezeichneten Kommunikation zwischen zwei Instanzen werden verwendet, um eine neuerliche Kommunikation einzuleiten.

Man in the Middle Attacke: Der Angreifer schaltet sich zwischen die Kommunikation von zwei Instanzen und täuscht sie so, dass von beiden angenommen wird sie würden miteinander kommunizieren.

Reflection Attacke: Eine Anfrage einer Instanz wird an diese zurückgeschickt. Dies kann zu Erfolg führen, wenn die Instanz die Anfrage selbst beantwortet oder die reflektierte Antwort als korrekt ansieht.

In Protokollen werden nun verschiedene Mechanismen verwendet um diese Angriff zu verhindern.

2.5.1 SSL und TLS

Das Transport Layer Security Protokoll (früher Secure Socket Layer - SSL) stellt Methoden zur Sicherung eines Kommunikationskanals zur Verfügung und ist im RFC5246 (vgl. [18]) definiert. TLS wird dabei zwischen Transportschicht und einem höherwertigen Protkoll (HTTP, FTP, SMTP usw.) eingesetzt.

Bei TLS können verschiedene Algorithmen zur Absicherung des Kanals eingesetzt werden. Dazu werden zwischen Server und Client so genannte Cipher Suits ausgehandelt. Der Client sendet dazu eine List von Cipher Suites die unterstützt werden. Der Server wählt eine dieser Suites. Ist keine Suite dabei, die der Server unterstützt, wird die Verbindung beendet. Diese Cipher Suites definieren für alle bei der Kommunikation relevanten Vorgänge einen Algorithmus. So kommt zum Beispiel für die asymmetrischen Vorgänge RSA, DSA oder Diffie-Hellman zum Einsatz. Als Hash-Funktionen könnten MD5 oder ein Mitglied der SHA Familie eingesetzt werden. Zur symmetrischen Verschlüsselung kommen Protokolle wie AES, 3DES oder RC4 in Frage.

TLS besteht weiters aus zwei Schichten (vgl. [25]):

In der untersten Schicht befindet sich das **TLS Record Protocol**. Dieser Teil von TLS setzt direkt auf die Transportschicht auf und verschlüsselt Daten für das darunter liegende Kommunikationsprotokoll (z.B.: TCP). Zu diesem Zweck werden symmetrische Algorithmen wie AES (siehe Abschnitt 2.1.4) oder TripleDES verwendet. Zur Übertragung größerer Datenmengen werden diese in Blöcke aufgeteilt und mittels CBC-Mode verschlüsselt. Ein Message Authentication Code wie HMAC und SHA1 (Abschnitt 2.3.3 und 2.3.1) dient zur Sicherstellung der Integrität.

Die gegenseitige Authentifizierzung der Kommunikationspartner und die Aushandlung von Schlüsselmaterial übernimmt das **TLS Handshake Protocol**. Dazu werden in einem speziellen Handshake die Zertifikate ausgetauscht (siehe Abbildung 2.23) und ein Rundenschlüssel für die symmetrische Verschlüsselung durch das Record Protokoll ausgehandelt. Zur Authentifizierung kommen X509 Zertifikate zum Einsatz. Diese werden mittels asymmetrischer Kryptographie (wie z.B.: RSA) signiert und überprüft. In Abschnitt 2.5.2

werden Zertifikate und die zugrundeliegende Infrastruktur näher erläutert. Weiters werden durch diesen Teil von TLS auch die zu verwendenden Algorithmen festgelegt. Dazu sendet der Client ein Client Hello an den Server. Dies enthält eine Zufallszahl, die Session ID und die Liste der unterstützten Cipher Suites. Der Server antwortet mit einem Server Hello (beinhaltet Zufallszahl, Session ID und die gewählte Cipher Suite). In diesem Schritt wird auch das Server Zertifikat übertragen. Vom Client kann nun ein Server Key Exchange Paket übertragen werden. Es dient der Erstellung von Schlüsselmaterial. Weiters wird vom Client mitgeteilt, dass ab jetzt die vereinbarte Cipher Suite verwendet wird. Der Server bestätigt dies und kann nun mit dem durch TLS gesicherten Datenaustausch beginnen (siehe Abbildung 2.23) (vgl. [25]).

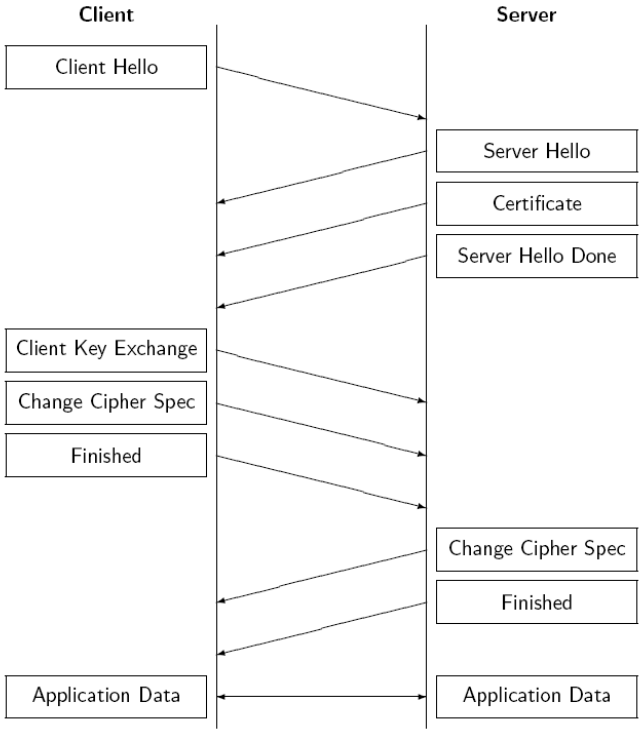

Abbildung 2.23: TLS Handshake mit RSA [25]

Weiters sind kleinere Protokollteile wie das Change Cipher Spec Protocol und das Alert Protocol definiert, die mit dem Handshake Protocol die zweite Schicht von TLS bilden (vgl. [18]).

2.5.2 Public Key Infrastrucktur

Eine Puplic Key Infrastruktur dient der Verwaltung von Zertifikaten für Schlüsselpaare bei asymmetrische Kryptographie. Es stellt sich das Problem sicherzustellen, dass ein gewisser Schlüssel wirklich zu einer bestimmten Person gehört. Als Lösung haben sich PKIs etabliert. Dabei gibt es einen vertrauten Dritten, der versichert, dass ein gewisser Schlüssel zu einer Person gehört. Dieser Dritte, auch Zertifizierungsstelle (engl. certificate authority, CA) genannt, stellt ein Zertifikat aus um zu bestätigen, dass ein Schlüssel zu einer Person gehört. Bei hirarchischen PKIs geschieht dies in Form einer Baumstruktur. Es gibt eine Wurzelinstanz (Root-CA), die ein Zertifikat über die darunterliegenden Instanzen

ausstellt. Diese wiederum stellen Zertifikate für weitere Instanzen aus (siehe Abbildung 2.24). Es wird nun zum Beispiel ein Zertifikat für A ausgestellt. Dies geschieht auf folgendem Weg (vgl. [47]):

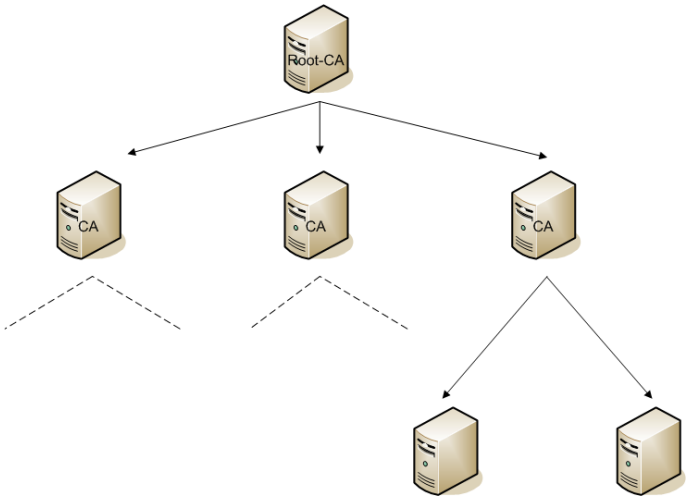

Abbildung 2.24: Hirarchische Public Key Infrastructure

$$Root - CA \rightarrow CA1 \rightarrow A$$

Will B nun diese Zertifikate überprüfen, wird zuerst das Zertifikat von der $ROOT-CA$ für $CA1$ überprüft und dann das Zertifikat von $CA1$ für A.

Ein anderes Konzept ist das "Web of Trust". Dabei stellt jede Instanz Zertifikate für andere Instanzen aus, die sie kennt (siehe Abbildung 2.25) (vgl. [47]).

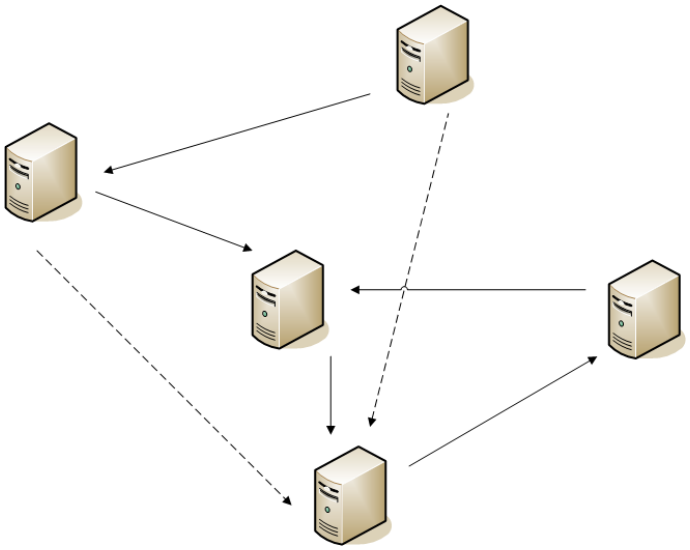

Abbildung 2.25: "Web of Trust" Public Key Infrastructure

Folgende Aufgaben sind Teil einer solchen Public Key Infrastructure (vgl. [47]):

- Ausstellen von Zertifikaten

- Widerrufen von Zertifikaten
- Speichern/Erneuern/Wiederherstellen von Schlüsselmaterial

Ausstellen von Zertifikaten: Die CA stellt Zertifikate über den öffentlichen Schlüssel von Benutzern aus. Je nach Anwendung kann dies auch die Generierung des Schlüsselsmaterials durch die CA beinhalten. Das Zertifikat sollte einen Gültigkeitszeitraum für den Schlüssel, den Identifikator des Austellers und des Benutzers, sowie weitere Angaben zur Verwendung enthalten. Beim Prozess der Generierung des Zertifikats muss sichergestellt sein, dass es sich bei dem Benutzer auch wirklich um den Benutzer handelt, für den er sich ausgibt. Weiters muss das Schlüsselmaterial und mit ihm auch das Zertifikat auf geheimem Wege an den Benutzer übertragen werden (vgl. [46]).

Widerrufen von Zertifikaten: Sollte ein privater Schlüssel verloren gehen oder für Dritte bekannt werden, muss dieser Schlüssel samt dem Zertifikat widerrufen werden. Da das Zertifikat nach Ausgabe durch die CA nicht mehr greifbar ist, muss dies durch andere Mittel bewerkstelligt werden. Dies wird meistens in Form einer Zertifikatssperrliste (engl. Certificate Revocation List, CRL) realisiert. Dabei werden widerrufene Zertifikate in diese Sperrliste eingetragen. Die Sperrliste muss daher für alle Personen einsehbar sein (vgl. [46]).

Speichern/Erneuern/Wiederherstellen von Schlüsselmaterial: Wie beim Ausstellen von Zertifikaten beschrieben, generiert die CA unter anderem auch das Schlüsselmaterial. Um notfalls den Schlüssel für Personen wiederherstellen zu können, muss ein Backup Mechanismus implementiert werden. Dabei ist es wichtig, dass Schlüssel geheim und sicher gespeichert werden und nur an authentifizierte Eigentümer weitergegeben werden. Die Erneuerung des Schlüsselmaterials betrifft den Gültigkeitszeitraum von Schlüsselmaterial. Sollte der Schlüssel in näherer Zukunft seine Gültigkeit verlieren, muss die Person mit einem neuen Schlüssel versorgt werden (vgl. [46]).

Im Folgenden wird ein solches System anhand eines Zertifikates für X.509 skizziert.

X.509 Zertifikat X.509 ist ein ITU-T Standard für eine hierarchische Public Key Infrastructure. Es wird in verschiedensten Anwendungen eingesetzt und ist zur Zeit bereits als Version 3 vorhanden. Im Folgenden wird der Aufbau eines Zertifikats und der Zertifikatssperrliste besprochen. Ein X.509 Zertifikat hat folgenden Inhalt (vgl. [46]) (siehe Abbildung 2.26):

- Version: Definiert die Version des verwendeten X.509 Zertifikats (Version 1,2 oder 3)
- Seriennummer: Eine eindeutige Nummer, die von der Zertifizierungsstelle für jedes erstellte Zertifikat vergeben wird.
- Signatur Algorithmus: Bezeichnet den Algorithmus, der zum Signieren verwendet wurde.
- Name der Zertifizierungsstelle
- Gültigkeitszeitraum: Beinhaltet das Datum, ab dem das Zertifikat gültig ist und das Datum, ab dem es seine Gültigkeit wieder verliert.
- Name des Benutzers

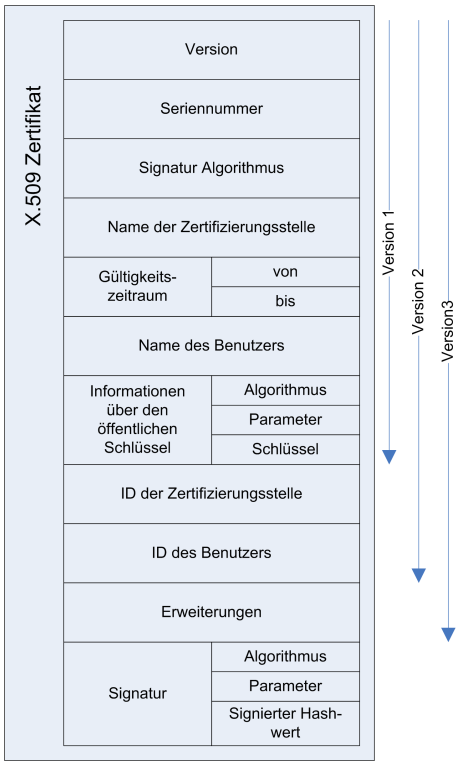

Abbildung 2.26: X.509 Zertifikat [46]

- Informationen über den öffentlichen Schlüssel des Benutzers: Beinhaltet den öffentlichen Schlüssel und Information über den Algorithmus, mit dem der Schlüssel verwendet werden soll.
- ID der Zertifizierungsstelle: Optionales Zusatzfeld (ab Version 2) mit eindeutiger ID für die Zertifizierungsstelle.
- ID des Benutzers: Optionales Zusatzfeld (ab Version 2) mit eindeutiger ID für den Benutzer.
- Erweiterungen: Ab Version 3 können zusätzliche Erweiterungen an ein X.509 Zertifikat angefügt werden.
- Signatur: Beinhaltet die Bezeichnung des Algorithmus, diverse Parameter und den Hast-Wert aller anderen Felder, der mit dem privaten Schlüssel der Zertifizierungsstelle signiert wurde.

Zusätzlich wird im Standard zu X.509 eine Zertifkatssperrliste definiert. Diese hat folgenden Aufbau (vgl. [46]) (siehe Abbildung 2.27):

- Signatur Algorithmus: Bezeichnet den Algorithmus, der zum Signieren der Zertifikatssperrliste verwendet wurde.
- Name der Zertifizierungsstelle
- Letztes Update: Datum des letzten Updates dieser Zertifikatssperrliste.

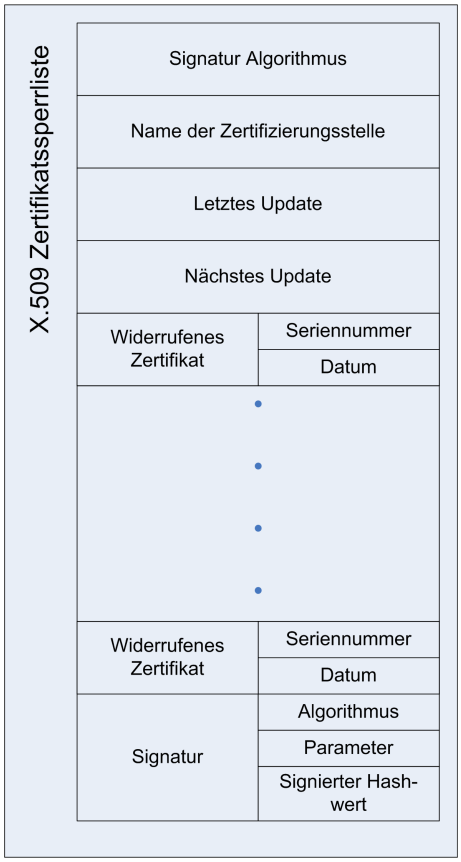

Abbildung 2.27: X.509 Zertifikatssperrliste [46]

- Nächstes Update: Datum des nächsten Updates der Zertifikatssperrliste.
- Widerrufene Zertifikate: Beinhaltet für jedes Zertifikat die Seriennummer und das Datum der Widerrufung.
- Signatur: Beinhaltet die Bezeichnung des Algorithmus, diverse Parameter und den Hash-Wert aller anderen Felder, der mit dem privaten Schlüssel der Zertifizierungsstelle signiert wurde.

X.509 ist aufgrund der Standardisierung weit verbreitet und liegt in unterschiedlichen Implementationen vor.

2.6 Zufallszahlen

Die Erzeugung von Zufallszahlen für kryptographische Zwecke nimmt einen wichtigen Stellenwert im Bereich der Kryptographie ein. Zufallszahlen sollen möglichst unvorhersehbar sein. Dies ist mit rein softwarebasierenden Algorithmen nicht möglich. Für richtige Zufallszahlen muss ein physikalischer Prozess, der unvorhersehbar ist, in die Generierung einbezogen werden. Dies geschieht meist durch Einrechnung hardwarebasierender zufälliger Parameter (z.B.: thermisches Rauschen). Solche Generatoren werden als nichtdeterministisch bezeichnet. Ist die Einbeziehung zufälliger Parameter nicht möglich, errechnet sich die Zufallszahl aus einem Startwert, der meist auf einer unvorhersehbaren Größe (z.B.:

CPU Temperatur, Benutzereingabe) basiert. Abhängig von diesem Startwert werden weitere Zufallszahlen generiert. Wird nochmals der selbe Startwert verwendet, führt dies wieder zu den selben Zufallszahlen. Diese Art der Generierung von Zufallszahlen wird als deterministisch bezeichnet. Solche Zufallszahlen werden weiters zur Unterscheidung echter Zufallszahlen als Pseudozufallszahlen bezeichnet. Da bei reinen Softwarealgorithmen keine Möglichkeit besteht, bei jeder Generierung zufällige physikalische Parameter einzubeziehen, sind diese immer deterministisch (vgl. [20]). Auch für diese Pseudozufallszahlen gelten folgende Eigenschaften:

- gleichmäßige Häufigkeitsverteilung der Werte
- kein erkennbarer Zusammenhang zwischen aufeinanderfolgenden Pseudozufallszahlen
- nicht periodisch

Ein Generator für Pseudozufallszahlen wird Pseudo Random Number Generator (PRNG) bezeichnet. Meist wird ein Startwert verwendet um weitere Zufallszahlen zu erzeugen. Der Startwert muss daher möglichst geheim bleiben. Probleme entstehen, wenn die Generierung der Zufallszahlen vorhersehbar ist oder sich Zufallszahlen wiederholen. Dies führte in der Vergangenheit bereits zu massiven Sicherheitsproblemen (vlg. [21]). Für die Generierung von Pseudozufallszahlen werden meist Blockchiffren oder Hash-Funktionen verwendet. Die richtige Verwendung dieser kryptograpischen Algorithmen wird in der NIST Special Publication 800-90 (vgl. [20]) beschrieben.

2.7 Anonymisierung mit kryptographischen Mitteln

In Zeiten zunehmender Vernetzung gewinnt die Anonymität in vielen Bereichen des digitalen Lebens immer mehr an Bedeutung. Es existieren bereits Lösungen die unterschiedliche Teilbereiche abdecken. Allen gemein ist, dass es sich dabei nie um eine ultimative Lösung handelt. Meist handelt es sich um einen Kompromiss zwischen erzielter Anonymität und dem zu betreibenden Aufwand.

2.7.1 Anonymisierung im Internet

Jeder Teilnehmer im Internet wird über eine Netzwerkadresse identifiziert. Diese Netzwerkadresse könnte zwar gefälscht werden, jedoch kann dann auch keine beidseitige Kommunikation zustande kommen. Der Internet Service Provider nimmt die Zuordnung von Netzwerkadresse zu realer Identität vor und kann diese auch auf behördlichen Wunsch offen legen. Weiters können Daten zu einer Netzwerkadresse gesammelt werden und daraus ein Nutzerprofil oder im schlimmsten Fall die Identität des Benutzers ermittelt werden. Diesem Problem haben sich einige Open Source Projekte angenommen. Es besteht die Möglichkeit einen Proxyserver zur Verschleierung der Identität zu verwenden. Dies erfordert ein hohes Level an Vertrauen an den Betreiber des Proxyservers. Andererseits exisitiert eine Form der Anonymisierung in Form der sogenannten Onion-Routing Netzwerke. Dabei werden die Datenpakete über eine Anzahl von Routern umgeleitet. Die verwendeten Router werden in kurzen Abständen gewechselt. Zusätzlich wird das Datenpaket in mehreren Schichten verschlüsselt. Zur Sicherung der Kommunikation kommt meist asymmetrische Kryptographie zum Einsatz. Jeder Knoten erhält das Paket verschlüsselt mit seinem öffentlichen Schlüssel. Das Paket wird vom Knoten entpackt. Dieser

findet im Paket die Adresse des nächsten Knotens und wiederum ein verschlüsseltes Paket. Das Paket wird an den nächsten Knoten weitergeleitet, der es wiederum mit seinem privaten Schlüssel entschlüsseln kann. Diese Prozedur wird bis zum Empfänger durchgeführt, der dann abschließend nochmals das Paket mit seinem privaten Schlüssel entpacken muss und die Nachricht lesen kann. Diese Verfahren wird in Abbildung 2.28 grob dargestellt (vgl. [38]).

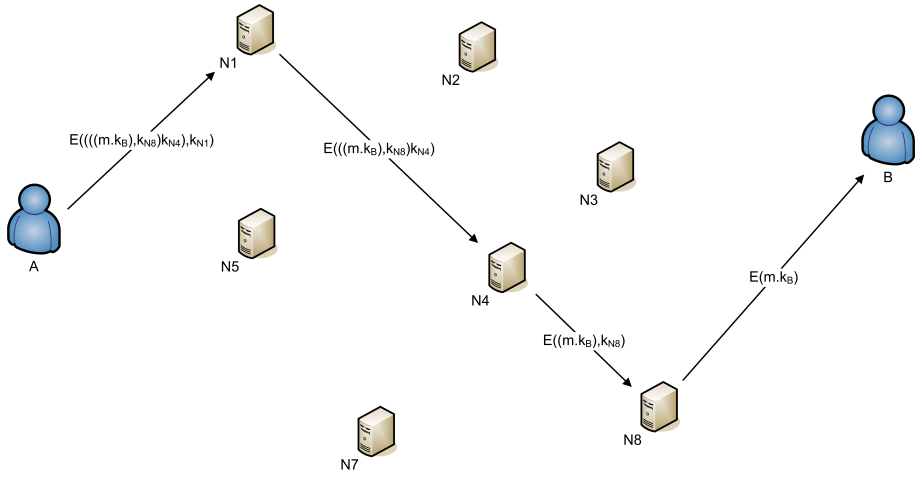

Abbildung 2.28: Onion Routing [38]

Natürlich berücksichtigen reale Implementationen wesentlich mehr Details um die Anonymität zu gewährleisten. Eine solche Umsetzung dieses Konzepts stellt das TOR-Netzwerk[1] dar.

2.7.2 Anonymisierung bei E-Voting

Ein weiteres Anwendungsgebiet in dem Kryptographie zur Anonymisierung elektronischer Abläufe eingesetzt wird ist das E-Voting. Beim E-Voting sollen alle Grundsätze der demokratischen Wahlen eingehalten werden. Dazu zählt das Recht jedes Wahlberechtigten wählen zu können, das die Wahl ohne Druck Dritter auf den Wählenden abgehalten werden kann und dass die Abgabe der Stimmen anonym erfolgt. Von David Chaum wurde 1981 ein Ansatz zur Umsetzung solcher elektronischer Wahlen entwickelt. Er basiert auf dem Einsatz einer dritten Instanz, eines sogenannten Mixes. Zusätzlich kommen asymmetrische Kryptographie und Signaturen zum Einsatz. Das Votum des Wählers wird mit dem öffentlichen Schlüssel des Wahllokals verschlüsselt. Weiters wird das Paket mit dem privaten Schlüssel des Wählers signiert und der öffentliche Schlüssel angefügt. Der öffentliche Schlüssel dient in weiterer Folge als Pseudonym des Wählers. Das in dieser Form verarbeitete Voting wird an den Mix gesendet. Der Mix entscheidet anhand einer Liste gültiger Pseudonyme über die Weitergabe des Votings an das Wahllokal. Das Wahllokal entschlüsselt das Votum und speichert es mitsamt des digitalen Pseudonyms in einer Liste. Mit dieser Methode ist gewährleistet, dass jeder Wähler nur einmal wählen kann, die Anonymität des Wählers gegeben ist und das Votum geheim gehalten wird (vgl. [49, 14]). Das von David Chaum entwickelte Verfahren ist in Abbildung 2.29 dargestellt. Außer dem Verfahren nach Chaum exisitieren noch weitere Konzepte für sichere elektronische

[1] http://www.torproject.org/index.html.de

Wahlen.

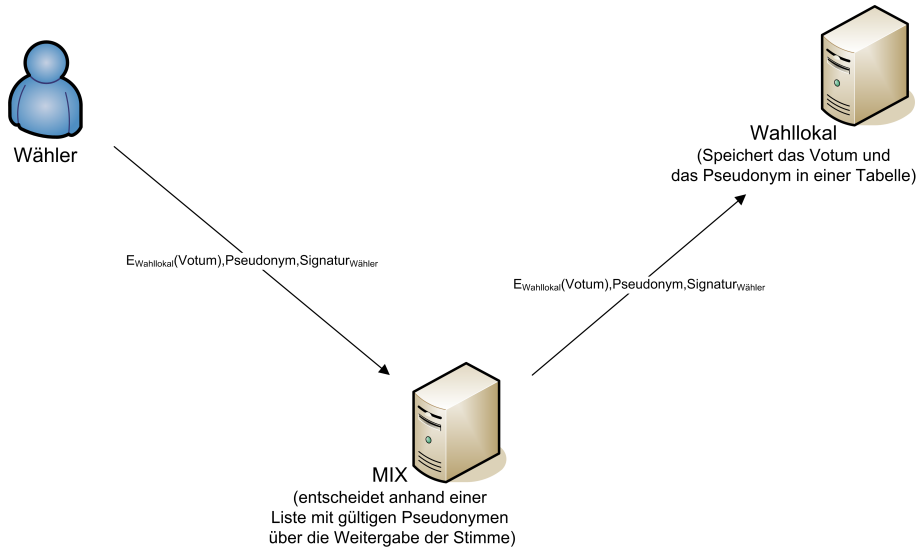

Abbildung 2.29: Mix Modell für E-Voting nach David Chaum [49]

2.7.3 Elektronisches Geld

Die Anforderungen an elektronisches Geld ähneln sehr den Anforderungen an elektronische Wahlen. Die Bank soll nicht sehen, wer welches Geld bei wem ausgibt. Zugleich soll der Verkäufer von der Echtheit des elektronischen Geldes überzeugt werden. Weiters muss verhindert werden, dass einzelne Münzen doppelt ausgegeben werden vgl. [36]. David Chaum hat auch für diese Probleme ein System entwickelt. Das von ihm entworfene System eCash basiert auf dem Konzept für elektronische Wahlen und ist im Folgenden grob beschrieben. Bei eCash kommen blinde Signaturen zum Einsatz. Dabei wird nicht preisgeben was signiert wird. Dies wird durch die multiplikativen Eigenschaften mancher asymmetrischer Algorithmen ermöglicht. So wird vom Kunden eine möglichst große Seriennummer s und ein Blendfaktor r gewählt. Beide Zahlen sollten möglichst zufällig sein. Der Blendfaktor wird mit dem öffentlichen Schlüssel der Bank per RSA verschlüsselt und abschließend mit der Seriennummer multipliziert (vgl. [13]):

$$m = s \cdot E_{Bank}(r) = s \cdot r^e \pmod{n}$$

Die enstandene Nachricht m wird an die Bank gesendet. Diese signiert die Nachricht und bucht einen vorher ausgehandelten Betrag vom Konto des Kunden.

$$c = S_{Bank}(m) = m^d \pmod{n}$$

Die signierte Nachricht wird wieder an den Kunden gesendet. Durch die multiplikativen Eigenschaften der RSA Verschlüsselung kann der Blendfaktor vom Kunden entfernt werden. Er erhält eine durch die Bank signierte Seriennummer y.

$$c \equiv m^d \equiv (s \cdot r^e)^d \equiv s^d \cdot r \pmod{n}$$

$$y \equiv c \cdot r^{-1} \pmod{n}$$

Dieser von der Bank signierte Scheck y kann bei einem Verkäufer eingelöst werden, welcher daraufhin die Signatur der Bank verifiziert und den Scheck an die Bank weiterleitet. Die Bank verifiziert nochmals die Signatur, überprüft in einer Datenbank ob der Scheck bereits einmal eingelöst wurde und bucht, falls der Scheck in Ordnung ist, das Geld auf das Konto des Verkäufers (vgl. [13]). Das blinde Signieren macht bei diesem Konzept die Verfolgung des Geldes durch die Bank unmöglich, da zum Zeitpunkt der Signatur für die Bank nicht erkennbar ist, um welche Seriennummer es sich handelt. Es kann jedoch verhindert werden, dass ein Scheck zweimal eingelöst wird und falls es zu einem Diebstahl beim Kunden kommen sollte, kann durch Offenlegung der Seriennummer ein Einlösen dieser Schecks verhindert werden.

Abhängig von der Größe der Transaktion existieren verschiedene weitere Konzepte zur Umsetzung von elektronischen Finanztransaktionen.

2.7.4 Zugriffsschutz bei RFID-Systemen

RFID-Tags werden immer günstiger. Aus diesem Grund dringen sie immer mehr in den bislang von Barcode-Systemen dominierten Markt ein. Ein Nachteil von RFID-Systemen ist dabei aber, dass die Tags auch ohne Sichtkontakt über mehrere Meter Entfernung gelesen werden können. Der Schluß liegt nahe, dass durch ein Netzwerk von RFID-Lesegeräten und anhand der ID eines Tags, eine Verfolgung von Personen durchgeführt werden kann. Um dies zu verhindern zielen diverse Strategien darauf ab, das Lesen eines Tags durch unbefugte Lesegeräte zu verhindern. Die sehr begrenzte Menge an Ressourcen auf einem solchen Tag begrenzen auch die möglichen Lösungen. Drei verschiedene Lösungsmodelle werden in (vgl. [53]) vorgestellt. Eines davon wird im Folgenden näher erläutert.

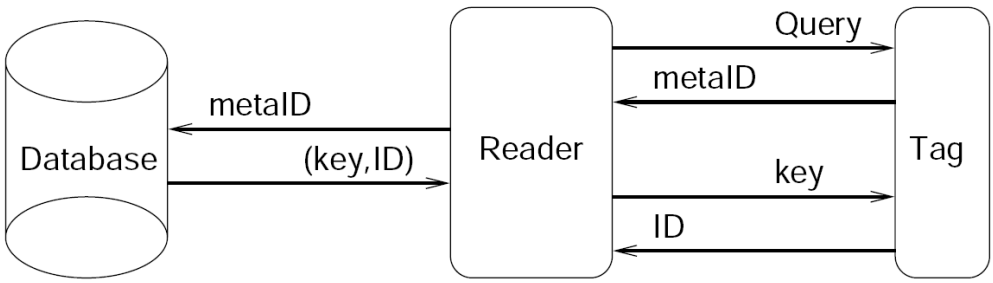

Abbildung 2.30: Hash-Locking Verfahren [53]

Beim Hash-Locking Verfahren gibt der Tag seine volle Funktionalität erst nach einer Identifikation durch das Lesegerät frei. Dabei erzeugt der Tag einen Hash-Wert seiner ID. Dieser Hash-Wert wird als $metaID$ bezeichnet. Auf Anfragen anwortet der Tag nur mit seiner $metaID$. Das Lesegerät ist nun mit einer Datenbank verbunden in der alle IDs und die zugehörigen $metaIDs$ gespeichert sind. Diese Daten müssen bei Herstellung des Tags dort abgelegt werden. Das Lesegerät überträgt die $metaID$ an die Datenbank und erhält als Antwort die ID des Tags. Die ID wird an den Tag übermittelt. Dieser errechnet daraus einen Hash-Wert und vergleicht ihn mit der gespeicherten $metaID$. Sind die beiden Werte gleich, ist der Tag entsperrt und steht für einen kurzen Zeitraum mit voller Funktionalität zur Verfügung (vgl. [53]). Der Ablauf dieses Verfahrens wird in Abbildung 2.30 dargestellt.

2.7.5 Anonymous Credentials

Neue Möglichkeiten elektronisch Geschäfte abzuwickeln haben zu der Notwendigkeit geführt, gewisse Eigenschaften aussagekräftig zu beweisen ohne die Anonymität im Internet zu untergraben. Das in der Literatur oft gebrauchte Beispiel zur Verdeutlichung dieser Notwendigkeit, ist der Kauf von Spirituosen über das Internet. Ein Kunde ($User$) möchte Spirituosen in einem Web-Shop erwerben. Dieser Verkäufer ($RelyingParty$ oder kurz RP) muss sich vor dem Verkauf vergewissern, dass der Kunde bereits das notwendige Alter für so einen Einkauf überschritten hat. Weiters muss er sicherstellen, dass die angegebene Versandadresse wirklich die Adresse des Kunden ist. Die Beweislast liegt dabei beim Kunden. Um im anonymen Internet solche Eigenschaften beweisen zu können, vertrauen beide auf eine dritte Instanz ($IdentityProvider$ oder kurz IDP). Diese dritte Instanz könnte eine staatliche Autorität oder Ähnliches sein. Der Benutzer fordert bei dieser Instanz nun eine Art "Pass" an. Dieser Pass enthält Daten wie Alter, Adresse, Geschlecht und Staatsangehörigkeit. Der Pass wird vom Identity Provider signiert. Will der Benutzer nun Spirituosen bei einem Händler einkaufen, muss er dort nur den signierten digitalen Pass vorweisen.

Das System in solcher Form untergräbt jedoch die Anonymität des Benutzers im Internet. Um sein Alter zu beweisen müsste er einen Pass vorlegen, in dem auch Name und Adresse dokumentiert sind. Weiters könnte der Händler den Pass weiterverwenden und sich als der Kunde ausgeben. Um diese und eine handvoll anderer Probleme in den Griff zu kriegen wurden eine Reihe von Algorithmen entwickelt, die es zulassen, dass der Kunde nur bestimmte Eigenschaften offenlegt. Andere Parameter bleiben verdeckt. Das EU-Projekt "Privacy and Identity Management for Europe"[2] (PRIME) beschäftigt sich mit der Entwicklung und Förderung solcher Algorithmen. Ein kurzes Beispiel für einen solche Algorihmus basierend auf dem Diskreten-Logartihmus-Problem wird im Folgenden anhand U-Prove erläutert (vgl. [8]):

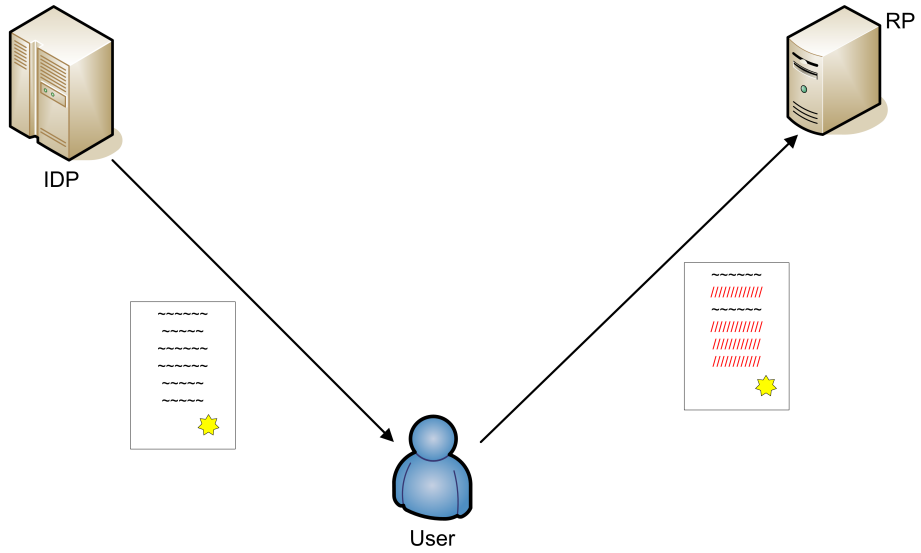

Abbildung 2.31: Anonymous Credentials [53]

U-Prove ist ein von Credentica entwickeltes und von Microsoft gekauftes System. Es

[2] https://www.prime-project.eu/

basiert auf dem Diskreten-Logarithmus-Problem. Der IDP verfügt über die Daten des Users und kann diese signieren. Es werden jedoch nicht die Daten selbst oder Hash-Werte der Daten signiert. Zuerst generiert der Identity Provider eine große Primzahl p und eine Anzahl von Generatoren $(g_1 \dots g_m)$ für die multiplikative Gruppe modulo p. Der Benutzer generiert eine Zufahlszahl α. α stellt in weiterer Folge den privaten Schlüssel des Benutzers dar und darf daher nicht an die Öffentlichkeit gelangen. Der Benutzer erhält vom IDP einen Generator g_m und die Primzahl p. Der führt daraufhin folgende Berechnungen durch:

$$A_m \equiv g_m^{\alpha} \pmod{p}$$

A_m wird an den Identity Provider gesendet (Der IDP kann α aus A_m nicht errechnen. Er müsste dazu das Diskrete-Logarithmus-Problem lösen). Dieser berechnet aus A_m, den restlichen Generatoren und den Parametern des Benutzers $(a_1 \dots a_{m-1})$ einen Wert der später signiert wird:

$$H \equiv g_1^{a_1} \cdot g_2^{a_2} \cdot g_3^{a_3} \cdot \ldots \cdot A_m \pmod{p}$$

Der Wert H wird abschließend signiert ($SIG_{IDP}(H)$) und dem Benutzer mitsamt der Generatoren und Parameter übergeben. Dieser kann nun verwendet werden um Parameter beim Verkäufer nachzuweisen. a_1 stellt in diesen Fall das Alter und a_2 die Adresse des Benutzers dar. Will der Benutzer diese zwei Eigenschaften nun beweisen, sendet er H, $SIG_{IDP}(H)$, die Generatoren, Parameter a_1, a_2 sowie den Rest $R \equiv g_3^{a_3} \cdot \ldots \cdot A_m \pmod{p}$ an den Verkäufer. Der Verkäufer überprüft nun die Signatur der Parameter indem folgende Berechnung durchgeführt wird:

$$H' \equiv g_1^{a_1} \cdot g_2^{a_2} \cdot R \pmod{p}$$

Entspricht nun H' den signierten Daten H, so sind diese Eigenschaften für den Verkäufer erwiesen. Um nun nachzuweisen, dass sich der Benutzer nicht einfach durch eine Replay Attacke (siehe Abschnitt 2.5) als jemand anderes ausgibt, muss er beweisen, dass er auch Kenntnis über die anderen Parameter und vor allem über α hat. Um diesen Beweis anzutreten generiert der Benutzer eine Reihe von Zufallszahlen $(w_1 \dots w_m)$ und berechnet daraus B analog zu R:

$$B \equiv g_3^{w_3} \cdot g_4^{w_4} \cdot g_5^{w_5} \cdot \ldots \cdot g_m^{w_m} \pmod{p}$$

B wird an den Verkäufer übertragen. Dieser anwortet mit einer Zufallszahl c. Daraus berechnet der Benutzer:

$$b_3 = c \cdot a_3 + w_3$$
$$b_4 = c \cdot a_4 + w_4$$
$$\vdots$$
$$b_m = c \cdot \alpha + w_m$$

Die Werte $b_3, b_4, \dots, b_m$ werden an den Verkäufer übertragen. Dieser errechnet nun:

$$R^c \cdot B \equiv g_3^{b_3} \cdot g_4^{b_4} \cdot \ldots \cdot g_m^{b_m} \pmod{p}$$

Auf diese Weise kann der Verkäufer nichts über die Parameter $a_3, a_4, \ldots, \alpha$ lernen, ist aber versichert, das der Benutzer die Parameter kennt.

Ein weiteres System, das von der EU im Zuge des PRIME Projekts gefördert wird und die selbe Zielsetzung wie U-Prove hat, ist IDEMix[3] von IBM.

[3] http://idemix.wordpress.com/

Kapitel 3

Konzepte zur Anonymisierung

3.1 Problemstellung

Die klassische Anonymisierung beschäftigt sich damit, alle personenrelevanten Daten zu entfernen und damit die Zuordnung zu realen Personen zu unterbinden. Für die Aufgabenstellung des BLIDS Systems ist es jedoch notwenig, die Daten zu einem späteren Zeitpunkt einander zuzuordnen. Daher kommt hier eher eine Pseudonymisierung in Frage. Bei der Pseudonymisierung wird ein Name (oder eine Bluetooth-Adresse) durch einen eindeutigen Identifikator ersetzt. Dieser Identifikator darf jedoch keine Rückschlüsse auf den Namen zulassen, muss jedoch auf lange Zeit eindeutig zugeordnet sein. Bei dieser Diplomarbeit bewegen wir uns zwischen Pseudonymiserung und Anonymisierung. Zum einen sollen die Bluetooth-Adressen für einen gewissen Zeitraum eindeutig einem einzigen Identifikator zurgeordnet werden. Außerhalb dieses Zeitraums soll die Zuordnung jedoch nicht mehr eindeutig sein. Die Begriffe Anonymisierung und Pseudonymisierung werden in Kapitel 1.4 näher erläutert.

3.1.1 Problem Langzeitverfolgung

Ist eine Verfolgung von Fahrzeugen über die Dauer einer Fahrt hin möglich, spricht man von Langzeitverfolgung. Dabei wäre zum Beispiel das Aufzeichnen und Zuordnen verschiedener Fahrten an verschiedenen Tagen möglich. Aus solchen Daten könnten Rückschlüsse auf die Gewohnheiten, den Arbeitsplatz oder den Wohnort des Fahrzeughalters errechnet werden. Im schlimmsten Fall könnte durch Abgleich mit anderen Datenbank die Identität des Fahrzeughalters bestimmt werden. Daher ist eine Langzeitverfolgung unbedingt zu verhindern.

BLIDS erfordert jedoch, dass ein Fahrzeug, welches von verschiedenen Sensoren erfasst wird, einen eindeutigen Identifikator erhält. Ohne diesen Identifikator ist eine Verkehrsflussanalyse nicht möglich. Eine weitere Anforderung ist aber, dass eine Langzeitverfolgung unter keinen Umständen möglich sein soll. Zu diesen Umständen zählt auch eine böswillige Benutzung des Anonymisierungssystems. So ist es zum Beispiel bei Kenntnis des Anonymisierungsverfahren möglich, eine gesuchte Person mit diesem Verfahren zu anonymisieren und den dadurch erhaltenen Identifikator in der Datenbank des Verkehrsflusserfassungssystems zu suchen. Eine weit zurückreichende Langzeitverfolgung wäre möglich. Ein Hindernis für eine einfache Lösung des Problems ist auch die verteilte Struktur des BLIDS Systems. Alle Sensorstationen müssen aus einer Bluetooth-Adresse den selben Identifikator berechnen können. Daher müssen alle Sensorstationen über das selbe Schlüsselmaterial

verfügen. Wird jetzt aber das Schlüsselmaterial kompromittiert, steht einer Langzeitverfolgung oder auch der Echtzeitverfolgung von Fahrzeugen nichts im Wege.

3.2 Möglichkeit zur Anonymisierung

3.2.1 Anonymisierung durch Zufallszahlen und Zeitfenster

Um die anonymisierte Verfolgung von Fahrzeugen sicherzustellen, aber keine Langzeitverfolgung zu ermöglichen und auch die Datensicherheit auf lange Zeit zu gewährleisten, bietet sich der Einsatz von Zeitfenstern an. Dabei gelten die den Adressen zugeordneten Pseudonyme nur für einen gewissen Zeitraum. Nach diesem Zeitraum ist es nicht mehr möglich einer Bluetooth-Adresse wieder das selbe Pseudonym zuzuweisen. Dazu werden Zeitfenster definiert, die dem Erfassungsbereich, in dem die Verkehrsflusserfassung durchgeführt werden soll, entsprechen. Dieser Erfassungsbereich kann zum Beispiel die Strecke zwischen zwei Sensorstationen darstellen. Benötigt ein Fahrzeug bei durchschnittlicher Geschwindigkeit eine Stunde für die Durchquerung dieser beiden Sensorstationen, wird das Zeitfenster mit einer Stunde festgelegt. In diesem Zeitfenster wird jedes Fahrzeug mit einer einzigen Zufallszahl (N_i) anonymisiert. Nach einer Stunde wird die Zufallszahl durch eine neue Zufallszahl (N_{i+1}) ersetzt. Die alte Zufallszahl wird verworfen. Somit ist es nicht möglich ein Fahrzeug länger als eine Stunde zu verfolgen. Ein Problem tritt zu den Zeitpunkten auf, wenn die Zufallszahl gewechselt wird. Für ein Fahrzeug das kurz vor dem Wechsel eine Sensorstation durchfährt und nach dem Wechsel von einer zweiten Station erfasst wird, kann keine Verkehrsflussanalyse durchgeführt werden. Um dieses Problem zu lösen, besteht die Möglichkeit die Zufallswerte in einem Art Ringspeicher mit begrenzter Größe abzulegen. In einem solchen Speicher könnten zum Beispiel vier Zufallszahlen gespeichert werden. Alle 2 Stunden kommt eine neue Zufallszahl hinzu und überschreibt die älteste im Ringspeicher vorhandene Zufallszahl. Somit sind immer die 4 aktuellsten Zufallswerte gespeichert. Wird nun ein Fahrzeug erfasst, wird es jeweils mit jedem der 4 Zufallszahlen anonymisiert. Es entstehen 4 anonymisierte Datensätze, die mit den bereits in der Datenbank gespeicherten Werten verglichen werden und eine Verkehrsflussanalyse auch über die Grenzen des Zeitfensters ermöglichen, aber keine Langzeitverfolgung zulassen. Für dieses Verfahren muss mindestens ein Ringspeicher mit 2 Zufallszahlen verwendet werden. Ringspeicher mit mehreren Werten bieten nur den Vorteil, dass der Wechsel der Zufallszahlen öfter durchgeführt werden kann. Dadurch wird eine Zufallszahl für wenigere Anonymisierungsvorgänge verwendet und das Zeitfenster kann kleiner angelegt werden. Eine grössere Anzahl von Zufallswerten bringt ansonsten keine Vorteile. Jedoch entsteht durch mehr Zufallswerte ein höheres Datenaufkommen. Zur Anoynmisierung kann eine Hash-Funktion aus Abschnitt 2.3 verwendet werden. Im einfachsten Fall wird die Bluetooth-Adresse (A) gepaddet, mit der Zufallszahl verknüpft und anschließend gehasht:

$$p_{A,i} = Hash(A \oplus N_i)$$

Beim Wechsel der Zufallszahl ensteht aus der gleichen Adresse ein anderer Hash-Wert:

$$p_{A,i+1} = Hash(A \oplus N_{i+1})$$

In der Datenbank kann dadurch später keine Verbindung zwischen den Werten $p_{A,i}$ und $p_{A,i+1}$ erkannt werden. Als Alternative kann der in Abschnitt 2.3.3 erwähnte MAC-

Algorithmus zum Einsatz kommen. Bei einem MAC-Algorithmus ist die Verwendung eines Schlüssels bereits vorgesehen. Dieser Schlüssel wird in den Hash-Wert eingebracht. Eine Reproduktion ist ohne Kenntnis über den Schlüssel nicht möglich. Die Zufallszahl N_i stellt in unserem Fall den Schlüssel für den MAC-Algorithmus dar. Die Einbringung der Zufallszahl in den Hash-Wert wäre damit bereits wissenschaftlich untersucht. Nachteilig ist der erhöhte Rechenaufwand bei der Verwendung eines MAC-Algorithmus. Eine Umsetzung mittels MAC wird im Folgenden mathematisch dargestellt:

$$p_{A,i} = MAC(A, N_i)$$
$$p_{A,i+1} = MAC(A, N_{i+1})$$

Dieses Konzept wird in Abbildung 3.1 näher erläutert.

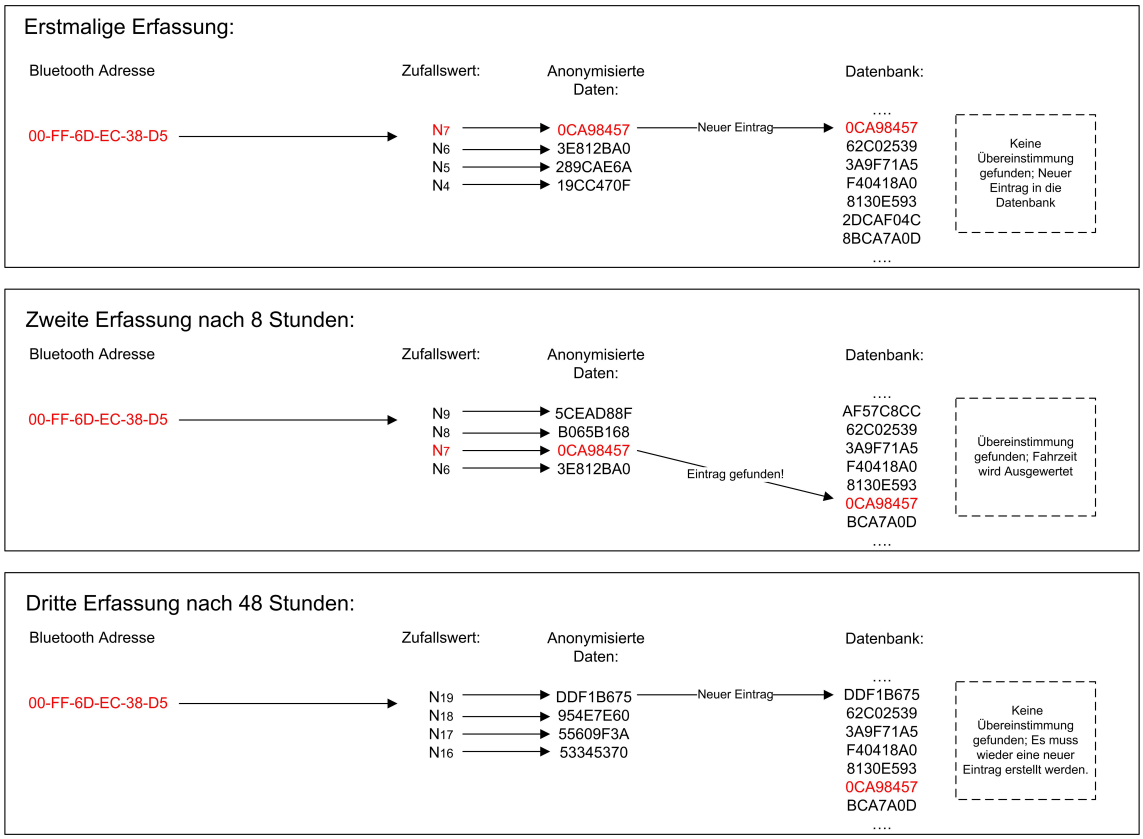

Abbildung 3.1: Anonymisierung mit Zeitfenster und Zufallszahlen (Nx)

3.2.2 Pseudonymisierung

Eine weitere Möglichkeit um eine datenschutzkonforme Anonymisierung durchführen zu können, ist eine Pseudonymisierung über eine dritte Instanz. Die Bluetooth-Adressen werden von den Sensorstationen an diese dritte Instanz übertragen. Die Übertragung muss dabei über einen gesicherten Kanal erfolgen. Diese Instanz weist jeder Adresse einen eindeutigen Identifikator (p) zu. Dieser Identifikator gilt auf unbestimmte Zeit und muss wieder über einen sicheren Kanal an die Sensorstationen übertragen werden. Die Zuweisung der Adressen zu Identifikatoren sollte über eine Einwegfunktion unter Einbeziehung

eines Schlüssels (k) erfolgen. Der Schlüssel darf nur dieser Instanz (dem Pseudonymisierungsserver) bekannt sein. Damit wird verhindert, dass aus dem Identifikator auf die Adresse rückgerechnet werden kann (Einwegfunktion) und die selbe Zuweisung von Dritten durchgeführt werden kann (geheimer Schlüssel). Ein Message Authentication Code (MAC) erfüllt diese Anforderungen (siehe Abschnitt 2.3.3). Eine Umsetzung mittels MAC könnte folgende Form haben:

$$p_A = MAC(A, k)$$

Problematisch ist, dass die Identifikatoren über einen langen Zeitraum gültig sind. Dies würde eine Langzeitverfolgung anhand bestimmter Bewegungsmuster des Fahrzeugs erlauben. Eine solche Lösung hat außerdem mit großen zu übertragenden Datenmengen zu kämpfen. Zum einem müssen die Adressen von den Sensorstationen und die Identifikatoren zurück übertragen werden. Zum anderen müssen Informationen zum Aufbau des sicheren Kanals ausgetauscht werden. Dies alles muss verschlüsselt erfolgen. Des Weiteren ist die Instanz, welche die Pseudonymisierung vornimmt, ein einzelner Angriffspunkt mit hohem Potential. Durch Kompromittierung dieser Instanz kann auf einfache Weise eine Langzeitverfolgung durchgeführt werden. Dieses System basiert auf einem Konzept zu elektronischen Wahlen [14].

Im Folgenden werden einige Konzepte vorgestellt. Mit diesen Konzepten ist eine Verkehrsflusserfassung im Sinne des Datenschutzgesetzes möglich. Zu großen Teilen basieren die Konzepte auf den beiden bereits vorgestellten grundsätzlichen Methoden zur Anonymisierung (siehe Abschnitt 3.2). Sie unterscheiden sich jedoch grundlegend in der Art wie diese Anonymisierung realisiert wird. So werden unterschiedliche Komponenten und Strukturen verwendet. Am wichtigsten sind jedoch die verschiedenen Eigenschaften, wie Sicherheit gegen Angriffe und die erzielte Performance.

3.2.3 Statistische Betrachtung

Bei den beiden vorgestellten Konzepten wird davon ausgegangen, dass ein Hash-Algorithmus wie SHA-1 oder SHA-256 (siehe Abschnitt 2.3) zur Anonymisierung der Bluetooth-Adressen verwendet wird. Ein solcher Algorithmus komprimiert einen meist großen Datenblock (bei SHA-256 512 Bit) zu einem kleineren Hash-Wert (in diesem Fall 256 Bit). Dabei muss es ab einer gewissen Anzahl von Hash-Vorgängen zu Wiederholungen bei den Hash-Werten kommen. Die Wahrscheinlichkeit auf Wiederholungen des selben Hash-Werts bei unterschiedlichen Bluetooth-Adressen zu stoßen ist in diesem Anwendungsgebiet jedoch nahezu 0. Viel wahrscheinlicher ist die Möglichkeit, dass durch die mehrfache Vergabe von gleichen Bluetooth-Adressen durch den Hersteller die Verkehrsflußanalyse verfälscht wird. Ein weiteres Problem stellen Fahrzeuge in denen sich mehrere Bluetooth-Geräte befinden (z.B.: Buse) dar. Diese Probleme müssen jedoch bei der Analyse am Datenbankserver gelöst werden.

3.3 Konzepte

3.3.1 Anonymisierung durch den Datenbankserver

Die Grundidee diese Konzeptes ist die Durchführung der Anonymisierung am Datenbankserver ohne zusätzliche Komponenten. Dabei wird die Anonymisierung mittels Zufallszahlen wie in Abschnitt 3.2 beschrieben angewandt. An den Sensorstationen werden keine

Änderungen umgesetzt. Die Fahrzeuge werden weiterhin erfasst, die Erfassung protokolliert und abschließend werden die gesammelten Daten an den Datenbankserver übertragen. Um ein Mitlesen der aufgezeichneten Daten zu verhindern, muss der Kanal gesichert werden. Zur Absicherung kommt SSL/TLS zum Einsatz. Der Datenbankserver übernimmt die Daten der Sensorstation und führt nun eine Anonymisierung mittels Hash-Funktion und Zufallszahlen durch. Die Zufallszahlen werden am Datenbankserver selbst mittels eines Zufallszahlengenerators generiert. Dabei ist besonderer Wert auf die Speicherung der Zufallszahlen zu legen. Zufallszahlen die nicht mehr gültig sind, müssen unbedingt verworfen werden.

Vorteile: Es werden nur wenige Komponenten benötigt. Anonymisierung und Datenbankserver können aus einer Hand betreut werden. Des Weiteren sind auf den Sensorstationen kaum Änderungen notwendig. Eine Absicherung des Übertragungskanals kann auch ohne Änderungen an der aktuellen Firmware realisiert werden. Die Anforderung an die Performance halten sich für die Sensorstationen in Grenzen. Zusätzlich zur Verschlüsselung mit einem symmetrischen Algorithmus, muss noch der Austausch der Sessionkeys mit asymmetrischer Kryptographie bewerkstelligt werden.

Nachteile: Am Datenbankserver sind sowohl Datenbank wie auch Zufallszahlen untergebracht. Dies bietet ein Ziel für potentielle Angriffe. Gelingt es die Zufallszahlen über einen längeren Zeitraum zu speichern, kann damit die Anonymisierung der bereits gespeicherten Adressen rückgängig gemacht und somit eine Langzeitverfolgung durchgeführt werden. Eine andere Möglichkeit das Anonymisierungssystem zu umgehen, ist die Abschaltung der Anonymisierungsfunktion und direkte Speicherung der Adressen in der Datenbank. Damit hat in erster Linie der Betreiber und in weitere Folge ein potentieller Angreifer zwei einfache Möglichkeiten das System außer Kraft zu setzen und Langzeitverfolgung durchzuführen. Durch Übernahme des Datenbankservers zu einem gewissen Zeitpunkt kann jedoch keine Langzeitverfolgung mit bereits aufgezeichneten Daten durchgeführt werden. Diese Annahme beruht auf dem Gedanken, dass alte Zufallszahlen in einem sicheren Verfahren gelöscht werden. Ist die Löschung der alten Zufallszahlen nicht sichergestellt, kann auch eine Langzeitverfolgung mit dem bereits bestehenden Datenbestand durchgeführt werden. Aus diesen Gründen kann dieses Konzept nicht empfohlen werden.

Angriffsmöglichkeiten

- Kontrolle über den Datenbankserver:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Eine Kontrolle über den Datenbankserver ermöglicht eine Langzeitverfolgung von Fahrzeugen ab dem Zeitpunkt der Übernahme (Forward Secrecy nicht gewährleistet!). Ein vorhandener Datenbestand kann jedoch nicht zur Langzeitverfolgung verwendet werden (Backward Secrecy).

- Kontrolle über eine Sensorstation:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Besteht Kontrolle über eine Sensorstation, so hat der Angreifer Einsicht in die Erfassungsvorgänge dieser einen Sensorstation. Weiters ensteht dadurch kein Nutzen.

- Einsicht in den Kanal zwischen Sensorstationen und Datenbankserver:

Dies wird möglich durch Brechen der Absicherung durch SSL/TLS oder das Wissen über den geheimen Schlüssel des Datenbankservers. Durch Einsicht in den gesicherten Kanal können alle Erfassungsvorgänge aufgezeichnet werden. Dies enstspricht dem Anlegen einer zweiten Datenbank (keine Forward Secrecy!).

3.3.2 Anonymisierung durch die Sensorstationen mit Server für die Zufallszahlen

Als Grundidee wird hier die Anonymisierung auf die Sensorstationen ausgelagert. Damit jedoch alle Sensorstationen zu jedem Zeitpunkt über die gleichen Zufallszahlen verfügen, kommt ein getrennter Server zum Einsatz der die Verteilung der Zufallszahlen an die Sensorstationen übernimmt. Dieser Server wird in weiterer Folge als Anonymisierungsserver bezeichnet. Bei diesem Konzept wird, wie auch im vorhergehenden Konzept, die Anonymisierung mit Hilfe von Zeitfenstern und Zufallszahlen aus Abschnitt 3.2 umgesetzt.

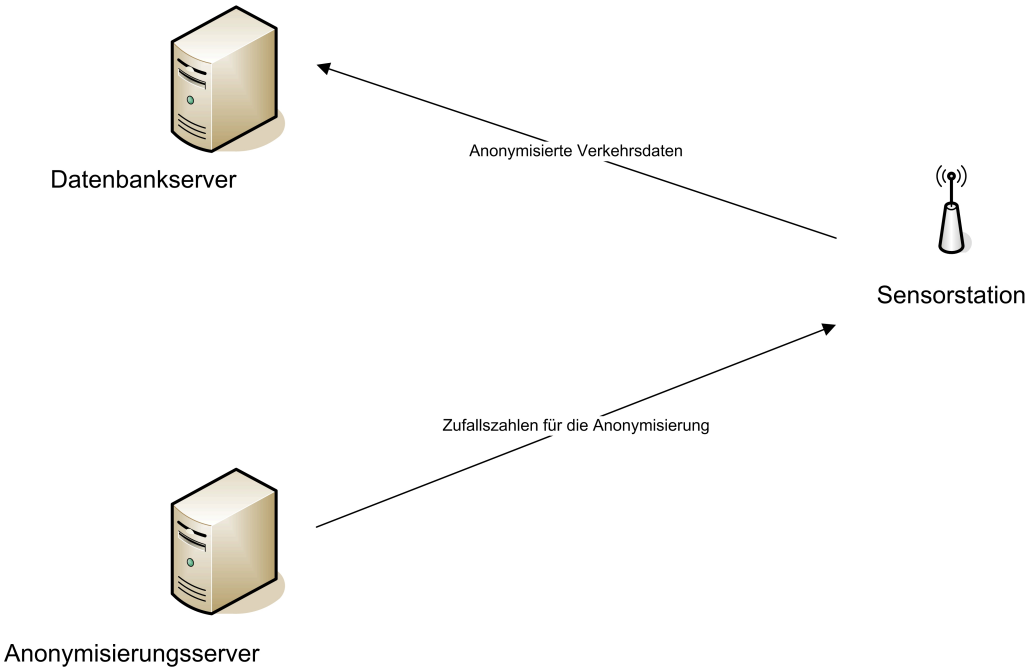

Abbildung 3.2: Anonymisierung durch die Sensorstationen mit Server für die Zufallszahlen

Die **Sensorstationen** erfassen die Fahrzeuge, die ihren Bereich durchqueren. Die Bluetooth Adressen werden mit den Zufallszahlen im Ringspeicher der Sensorstationen und einer Hash-Funktion anonymisiert. Die gesammelten Daten werden in weiterer Folge an den Datenbankserver gesendet. Da es sich bereits um anonymisierte Daten handelt, ist eine sichere Verbindung zum Datenbankserver nicht notwendig. Dies bietet jedoch einen zusätzlichen Sicherheitsgewinn.

Der **Datenbankserver** empfängt die anonymisierten Daten von der Sensorstation, vergleicht diese Daten mit den Daten die sich bereits in der Datenbank befinden und führt eine Verkehrsflussanalyse durch. Zudem versucht der Datenbankserver die Qualität der Daten durch statistische Methoden zu verbessern.

Der **Anonymisierungsserver** erzeugt Zufallszahlen und übermittelt diese am Ende jedes Zeitfensters an alle Sensorstationen. Die Zufallswerte werden am Anonymisierungs-

server nicht gespeichert. Wichtig ist, dass die Zufallswerte an die Sensorstationen über einen sicheren Kanal übertragen werden.

Zur Absicherung der Kommunikation zwischen Anonymisierungsserver und Sensorstationen oder optional zwischen Sensorstationen und Datenbankserver, kann SSL/TLS und eine Public Key Infrastructure wie in Abschnitt 2.5.2 dargestellt zum Einsatz kommen.

Vorteile: Durch die Aufteilung des Anonymisierungsvorgangs auf 3 Komponenten und dem Verwerfen der alten Zufallswerte kann keine Langzeitverfolgung durchgeführt werden. Nach dem Ablauf des Zeitfensters kann kein Fahrzeug den Werten in der Datenbank zugeordnet werden.

Ein weiterer Vorteil ist, dass die Anonymisierung von den Sensorstationen übernommen wird. Dadurch wird verhindert, dass eine Komponente des Systems übermäßig mit der Anonymisierung vieler Fahrzeuge überfordert wird. Der Aufbau des sicheren Kanals mit dem Anonymisierungsserver erfolgt in den Abständen der definierten Zeitfenster und fordert die Sensorstation nur kurzfristig. Pro Messung muss die Sensorstation zu den Mess- und Übertragungsaufgaben nur die Anonymisierung der Adresse bewerkstelligen. Bei einem geplanten Aufkommen von maximal bis zu 100 Fahrzeugen pro Minute und Sensorstation stellt dies für die meisten Systeme keine Hürde dar.

Der Datenbankserver (und somit der Auftraggeber) hat keinerlei Zugriff auf die Zufallszahlen für die Anonymisierung und kann daher auch keine Zuordnung von realen Fahrzeugen zu Datenbankeinträgen treffen. Dies trifft auch zu, wenn sich die Fahrzeuge im noch gültigen Zeitfenster befinden.

Nachteile: Es wird eine dritte Komponente benötigt. Zusätzlich zu Sensorstation und Datenbankserver muss ein Anonymisierungsserver vorgesehen werden. Im besten Fall ist dieser Anonymisierungsserver bei einem vertrauenswürdigen Dritten untergebracht. Auf jeden Fall sollte sichergestellt werden, dass die Kontrolle über Datenbank- und Anonymisierungsserver nicht von einer einzelnen Person wahrgenommen wird. Durch die Aufteilung der Kompetenz wird erreicht, dass die Zugriffsmöglichkeiten auf Zufallszahlen und Datenbank voneinander getrennt sind. Somit kann keine Einzelperson das System nutzen um eine Langzeitverfolgung von Fahrzeugen durchzuführen.

Ein erhöhtes Datenaufkommen zwischen Datenbankserver und Sensorstation ist zu erwarten, da Teile eines Datensatzes mindestens doppelt übertragen werden müssen. Dies ist jedoch abhängig von der gewählten Anonymisierungsfunktion. Zusätzlich zu den normalen Messdaten muss mindestens die doppelte Bitlänge des Hash-Werts der Hash-Funktion mitübertragen werden. Jedoch erspart man sich die Übertragung der Bluetooth-Adresse (48Bit). Soll der Kanal zwischen Sensorstation und Datenbankserver zusätzlich verschlüsselt werden, ist mit einem noch höherem Datenaufkommen auf diesem Kanal zu rechnen. Abhängig von der Art der Datenverbindung (bei den ersten Sensorstationen kommt GPRS als Verbindung zum Datenbankserver zum Einsatz) kann dies zu Einschränkungen führen.

Angriffsmöglichkeiten

- Kontrolle über den Datenbankserver:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Ohne weitere Kontrolle über Sensorstation oder Datenbankserver entsteht daraus

kein Nutzen. Bei Kenntnis der Zufallszahlen kann eine Langzeitverfolgung durchgeführt werden. Dies setzt die Kontrolle über Anonymisierungsserver oder Sensorstation voraus.

- Kontrolle über eine Sensorstation:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Besteht Kontrolle über eine Sensorstation, so hat der Angreifer Einsicht in die zur Zeit gültigen Zufallszahlen. Besteht Zugang zum Datenbankserver, kann mit den bekannten Zufallszahlen eine Langzeitverfolgung durchgeführt werden.

- Kontrolle über den Anonymisierungsserver:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Durch Kontrolle über den Anonymisierungsserver verfügt der Angreifer über die aktuellen und alle zukünftigen Zufallszahlen (keine Forward Secrecy!). Bei zusätzlicher Kontrolle über die Datenbank besteht die Möglichkeit zur Langzeitverfolgung.

- Einsicht in den Kanal zwischen Sensorstationen und Anonymisierungsservers:

  Dies wird möglich durch Brechen der Absicherung durch SSL/TLS oder das Wissen über den geheimen Schlüssel einer Sensorstaion. Der Angreifer erlangt dadurch Kenntnis über die momentan aktuellen Zufallszahlen. Besteht Kontrolle über den Datenbankserver kann für die gesammelten Zufallszahlen einen Langzeitverfolgung durchgeführt werden. Abhängig von der Funktion zur Generierung der Zufallszahlen kann auf zukünftige Zufallszahlen geschlossen werden (keine Forward Secrecy!).

3.3.3 Anonymisierung durch die Sensorstationen

Der Grundgedanke bei diesem Konzept ist die Abhandlung der Anonymisierung auf den Sensorstation. Die Daten würden nach einer Erfassung sofort anonymisiert werden und nur in anonymisiert Form an den Datenbankserver weitergegeben werden. Bei diesem Konzept werden nur zwei Komponenten benötigt: Sensorstationen und Datenbankserver. Beide Komponenten sind im BLIDS System bereits vorhanden.

Die **Sensorstationen** führen die Anonymisierung durch. Dazu verfügen sie über die notwendigen Zufallszahlen. Zukünftige Zufallszahlen werden von den Sensorstationen selbst generiert. Dazu ist ein Verfahren möglich, dass aus einem Initialwert eine weitere zufällige Zahl generiert. Initialwert wäre in diesem Fall die momentane Zufallszahl. Solche Algorithmen existieren in der Form von Pseudo Random Number Generators (PRNG) und sind im Standard NIST SP 800-90 [20] spezifiziert. Dieser Vorgang muss auf allen Sensorstationen zur gleichen Zeit stattfinden. Zusätzlich müssen alle Sensorstationen über den selben Initialwert verfügen.

Der **Datenbankserver** übernimmt in diesem Konzept keine besondere Rolle. Er empfängt die bereits anonymisierten Daten von den Sensorstationen und führt mit Hilfe dieser Daten eine Verkehrsflussanalyse durch. Zusätzlich kann die Verbindung zwischen Sensorstationen und Datenbankserver noch verschlüsselt werden. Dazu wäre jedoch eine Public Key Infrastructure nötig, die für den restlichen Anonymisierungsvorgang nicht benötigt wird.

Vorteile: Am Datenbankserver sind keine Änderungen umzusetzen. Weiters sind außer den Sensorstationen keine weiteren Komponenten notwendig und auch hier hält sich der

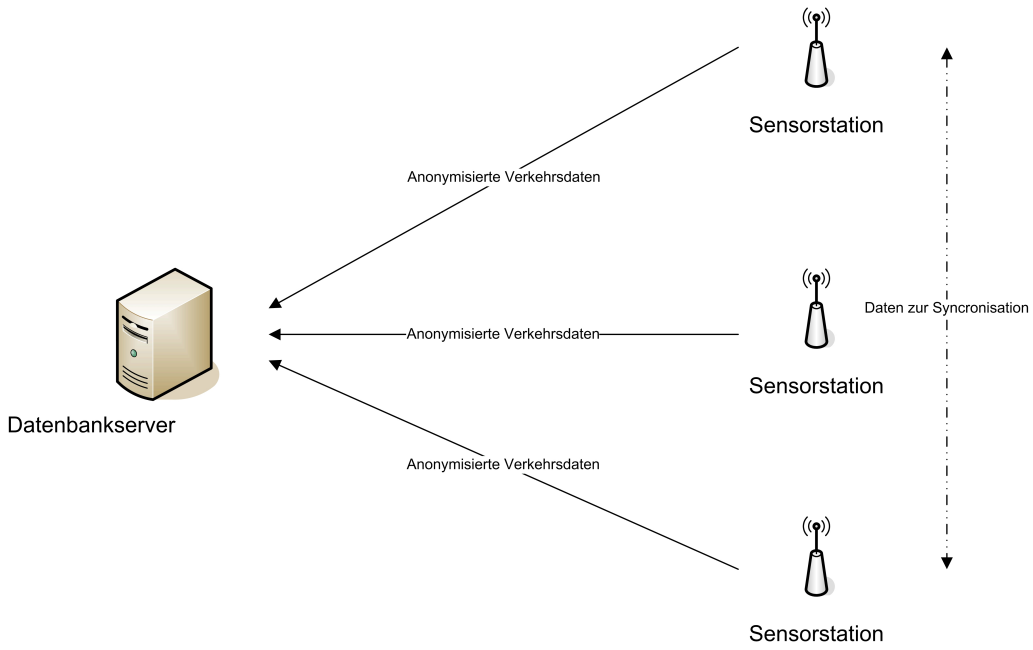

Abbildung 3.3: Anonymisierung durch die Sensorstationen

Änderungsaufwand in Grenzen. Zudem basiert die Anonymisierung nicht auf einer zentralen Stelle, wie bei den bisher besprochenen Konzepten. Ein Ausfall einer Sensorstation würde das System nicht beeinflussen. Zusätzlich bleiben alle sicherheitsrelevanten Daten (Zufallszahlen, Bluetooth-Adressen) immer auf den Sensorstationen und werden nicht an andere Instanzen übertragen.

Nachteile Die Synchronisation der Sensorsationen ist schwierig umzusetzen. Jede Sensorstation müsste über den selben aktuellen Stand an Zufallszahlen verfügen. Da die Zufallszahlen als Initialwerte für die Generierung weiterer Zufallszahlen dienen, muss zu jedem Zeitpunkt jede Sensorstation über den selben Status verfügen. Wichtig in diesem Fall ist auch, dass die Uhren aller Sensorstationen aufeinander abgestimmt sind. Um bei der Installation von zusätzlichen Sensorstationen diese auch mit den aktuellen Initialwerten zu versorgen, müssen diese Initialwerte bekannt sein. Eine Lösung für diese Problem muss überlegt werden.

Ein weiterer Nachteil ist, dass sobald eine Zufallszahl bekannt wird, alle nachfolgenden Zufallszahlen mit dem PRNG erzeugt werden können. Dieses Problem wird durch die dezentrale Situation der Sensorstationen noch zusätzlich verstärkt. So existiert nicht nur ein Angriffspunkt sondern viele. Jedoch gibt schon eine kompromitierte Sensorstation das gesamte System preis. Ein Wechsel der Zufallszahlen ist in so einem Fall nur schwer möglich. Dazu müssten Mechanismen und wiederum eine Public Key Infrastructure implementiert werden.

Angriffsmöglichkeiten

- Kontrolle über den Datenbankserver:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Durch Kontrolle über den Datenbankserver ensteht dem Angreifer kein Nutzen.

- Kontrolle über eine Sensorstation:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Besteht Kontrolle über eine Sensorstation, so hat der Angreifer Einsicht in die zur Zeit gültigen Zufallszahlen. Mithilfe dieser Zufallszahlen können zukünftige Zufallszahlen generiert werden. Eine vorhanden Adresse kann anonymisiert und mit der Datenbank abgeglichen werden. Ein Angriff auf eine Sensorstation ist vermutlich einfacher zu realisieren, als auf einen Server. Durch die exponierte Lage der Sensorstation ist ein physischer Zugriff durch einen Angreifer möglich. Wohingegen Server sich meist in einer gesicherten Umgebung befinden.

3.3.4 Pseudonymisierung auf getrenntem Server

Im Gegensatz zu den bisher vorgestellen Konzepten, basiert diese auf der Methode zur Pseudonymisierung aus Abschnitt 3.2. Dabei wird die Bluetooth-Adresse an eine dritte Instanz übertragen, welche diese durch einen Identifikator ersetzt und an die Sensorstation zurücksendet. Die Sensorstation ersetzt die Bluetooth-Adresse dann durch diesen Identifikator.

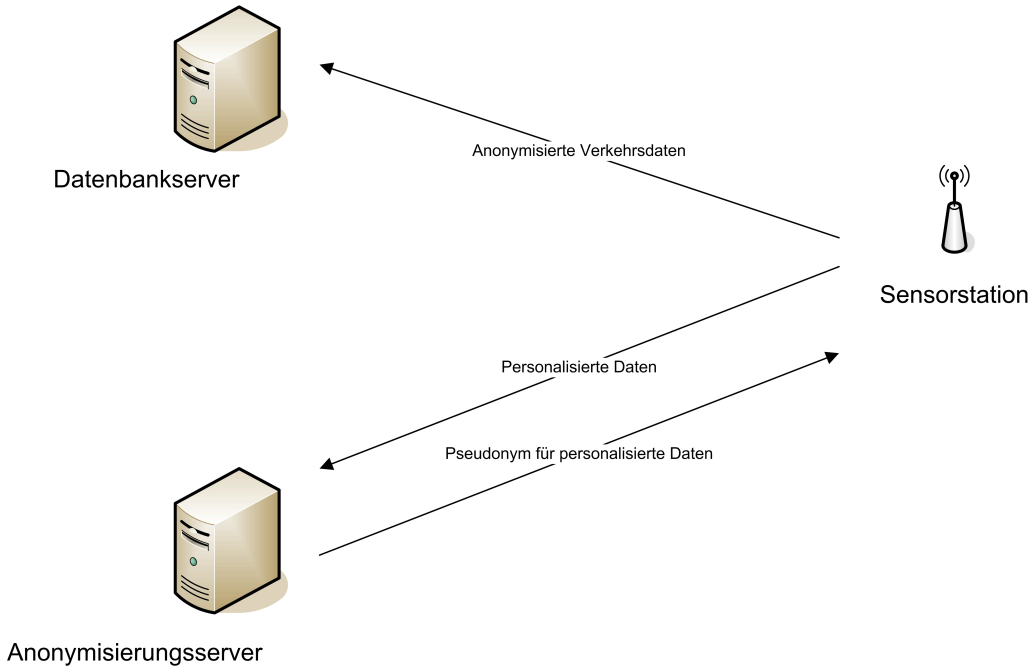

Abbildung 3.4: Pseudonymisierung

Die **Sensorstation** führt die Messung durch. Aus den Messdaten werden die personalisierten Daten entnommen. Diese werden über einen sicheren Kanal an den Pseudonymisierungsserver übertragen. Dieser sichere Kanal wird mittels TLS (Abschnitt 2.5.1) hergestellt. Für das Schlüsselmanagement kommt eine Public Key Infrastructure zum Einsatz. An Stelle der personalisierten Daten im Datensatz wird später das vom Pseudonymsierungsserver erhaltene Pseudonym eingesetzt. Anschließend werden die Daten gesichert per TLS an den **Datenbankserver** übertragen. Dieser führt mit Hilfe der Daten die Verkehrsflussanalyse durch. Der **Pseudonymisierungserver** erhält seine Daten von der Sensorstation und generiert ein eindeutiges Pseudonym. Dies kann durch einen

MAC-Algorithmus, wie in Abschnitt 2.3 vorgestellt, geschehen. Das Pseudonym wird anschließend an die Sensorstation zurück übertragen.

Vorteile Ein eindeutiger Vorteil dieses Systems ist die Ersetzung der Adresse durch ein Pseudonym auf einer zusätzlichen Komponente. Die Ermittlung eines Pseudonyms für eine Adresse erfolgt am besten unter Einbeziehung eines MAC und eines Schlüssels. Der Wechsel des Schlüssels ist zwar in periodischen Abständen sinnvoll, es existiert jedoch keine Notwendigkeit dies in kurzen Abständen durchzuführen. Die Geheimhaltung des Schlüssel ist für das Funktionieren natürlich von essentieller Wichtigkeit.

Nachteile Es ist eine zusätzliche Komponente in Form des Pseudonymisierungsservers notwendig. Die Kanäle zwischen Sensorstation und Pseudonymisierungsserver und zwischen Sensorstation und Datenbankserver müssen per TLS gesichert werden. Fehlt die Sicherung zwischen Sensorstation und Pseudonymisierungsserver, kann das Pseudonym direkt der Bluetooth Adresse zugeordnet werden. Bei jeder Messung muss die Sensorstation einen Kanal zu diesem Pseudonymisierungserver herstellen und ein Pseudonym für die Bluetooth-Adresse anfordern. Dies erzeugt einen hohen Kommunikationsoverhead. Abhängig von der Art der Datenverbindung (bei den ersten Sensorstationen kommt GPRS als Verbindung zum Datenbankserver zum Einsatz) und der Leistung der Sensorstationen kann dies zu Einschränkungen führen. Eine bessere Lösung wäre es, Bluetooth Adressen zu sammeln, und gesammelt an den Pseudonymisierungsserver zu schicken.

Angriffsmöglichkeiten

- Kontrolle über den Datenbankserver:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Ohne weitere Kontrolle über Pseudonymisierungsserver entsteht daraus kein Nutzen. Bei Kontrolle über den Pseudonymisierungsserver kann eine Langzeitverfolgung durchgeführt werden. Vorhandene Adressen können dann am Pseudonymisierungsserver in Pseudonyme umgewandelt werden. Diese Pseudoyme müssen dann mit der Datenbank abgeglichen werden.

- Kontrolle über eine Sensorstation:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Fahrzeuge im Bereich der Sensorstation können erfasst werden. Pseudonym kann so einem Fahrzeug zugeordnet werden. Falls weiters Kontrolle über den Datenbankserver besteht, kann eine Langzeitverfolgung für Fahrzeuge, die diese Sensorstation durchquert haben, durchgeführt werden.

- Kontrolle über den Pseudonymisierungsserver:

  Dies wird möglich durch Umgehung der Sicherheitsfunktionen des Betriebssystems. Durch Kontrolle über den Pseudonymisierungsserver ist Funktion zur Pseudonymisierung ist bekannt. Mit den Anforderungen der Senorstationen kann eine grobe Zuordnung von Fahrzeugen zu Sensorstationen durchgeführt werden. Eine weitere Kontrolle über den Datenbankserver ermöglicht eine Langzeitverfolgung.

- Einsicht in den Kanal zwischen Sensorstationen und Pseudonymisierungsserver:

  Dies wird möglich durch Brechen der Absicherung durch SSL/TLS oder das Wissen über den geheimen Schlüssel einer Sensorstation. Anhand der Zuordnung von Adressen zu Sensorstationen kann eine zweite Datenbank aufgebaut werden. Besteht Kontrolle über den Datenbankserver kann für die gesammelten Pseudonyme und Bluetoothadressen einen Langzeitverfolgung durchgeführt werden.

3.3.5 Anonymisierung mit Hilfe eines Secure Environment

Viele der oben behandelten Konzepte können durch den Einsatz eines Secure Environments erweitert werden. Als Secure Enviroments werden Systeme bezeichnet, bei welchen auf die gespeicherten Daten nur über festgelegte Schnittstellen zugegriffen werden kann. Beispiele für Secure Environments sind Smartcards oder Trusted Platform Modules (TPM). Ein Secure Environment kann entweder die Generierung und Verteilung der Zufallszahlen oder die Anonymisierung selbst übernehmen.

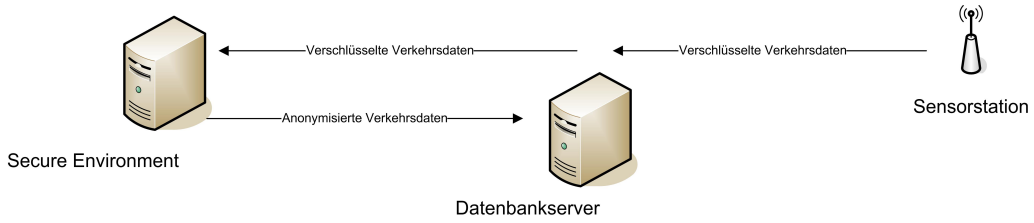

Abbildung 3.5: Anonymisierung mittels Secure Environment

Bei einer Ausführung wird die Anonymisierung im Secure Environment durchgeführt. Dazu werden die Messdaten dorthin übertragen und mit geeigneten Algorithmen in Kombination mit den Zufallszahlen anonymisiert. Nach Abschluss der Anonymisierung werden die anonymisierten Daten wieder an den Datenbankserver übertragen.

Die zweite Möglichkeit ähnelt der Anonymisierung mit Anonymisierungsserver. Dabei werden im Secure Environment Zufallszahlen generiert, die dann über einen sicheren Kanal an die Sensorstationen übertragen werden. Der Datenbankserver hat dabei keine Einsicht auf diesen sicheren Kanal.

Vorteile Ein großer Vorteil beim Einsatz von Secure Environments ist der beschränkte Zugriffe auf die Inhalte im Speicher. So ist es von außen nicht möglich an die Zufallszahlen zu gelangen oder in eine Anonymisierung einzugreifen. Zur Kommunikation müssen eigene Schnittstellen realisiert werden, welche die notwendigen Daten an die Außenwelt weitergeben. Bei Übertragung der Zufallszahlen an die Sensorstationen fällt dieser Vorteil zum Teil weg, da die Zufallszahlen zusätzlich den Sensorstationen bekannt sind.

Nachteile Smartcards sind in ihrer Leistung beschränkt. Bei einer geplanten Auslastung von maximal bis zu 100 Fahrzeugen pro Sensorstation und Minute, können bereits die Daten einer Sensorstation ein solches System an die Grenzen der Leistungsfähigkeit bringen. Dabei darf nicht nur der reine Anonymisierungsvorgang betrachtet werden, auch die Übertragung der Daten über eine sicheren Kanal und die Weitergabe an den Datenbankserver belasten zusätzlich. Die Einbindung von Trusted Computing wurde aus Komplexitäts- und Kompatibilitätsgründen nicht angedacht.

3.3.6 Zusammenfassung der Vor- und Nachteile der Konzepte

	Anonymisierung durch Datenbankserver	Anonymisierung durch Sensorstationen mit Server für Zufallszahlen	Anonymisierung durch die Sensorstationen
Sicherheit	-	+	+
Aufwand bei der Umsetzung	+	-	+
Anforderungen an die Leistung der Einzelkomponenten	-	+	-

	Anonymisierung mittels Secure Environment	Pseudonymisierung auf getrenntem Server	
Sicherheit	+	+	
Aufwand bei der Umsetzung	-	-	
Anforderungen an die Leistung der Einzelkomponenten	-	+	

Kapitel 4

Umsetzung

Als geeignete Mischung aus Sicherheit, Performance und einfacher Realisierung hat sich eine Anonymisierung durch die Sensorstationen herausgestellt. Basierend auf dem Konzept aus Abschnitt 3.3.3 wurde eine umsetzbare Lösung für die anonymisierte Verkehrsflussanalyse erarbeitet. Zusätzlich zum bereits grob erklärten Konzept, kommt eine Public Key Infrastructure zum Einsatz, die für den sicheren Austausch der Daten sorgt. Als Alternative wurde noch das Konzept mittels Pseudonymisierung detaillierter ausgearbeitet.

4.1 Anonymisierung durch die Sensorstationen

4.1.1 Anonymisierung

Wie bereits in Abschnitt 3.3.3 erläutert, wird die Anonymisierung von den Sensorstationen durchgeführt. Diese müssen dazu über die aktuell gültigen Zufallszahlen verfügen. Die Messwerte werden mit diesen Zufallszahlen verknüpft und in weiterer Folge mit einer Hash-Funktion anonymisiert. Ohne Kenntnis der Zufallszahlen ist es nicht möglich auf die Bluetooth Adressen rückzuschließen.

Die Anzahl der Zufallszahlen wird mit zwei festgelegt. Dies garantiert, dass die Fahrtstrecke von Fahrzeugen, die am Ende eines Wechselintervalls erfasst werden, weiter verfolgt werden kann (siehe Abbildung 4.4). Zum anderen bedeuten nur zwei Zufallszahlen wiederum einen geringeren Datenoverhead bei der Übertragung der Zufallszahlen an den Datenbankserver. Außerdem sind verhältnismäßig weniger Rechenschritte notwendig, da jede Adresse nur zwei Mal gehashed werden muss (Im Gegensatz zu n Zufallszahlen, wo jede Adresse n mal gehashed werden müsste.).

Zur Anwendung der Hash-Funktion auf die Bluetooth Adresse wurden zwei Verfahren ermittelt. Das erste Verfahren basiert auf der Verknüpfung der Bluetooth (BT) Adresse mit der Zufallszahl (R) durch XOR und ergibt somit das anonymisierte Ergebnis (C). Die Länge der Zufallszahl sollte dabei möglichst der Blocklänge der Hash-Funktion entsprechen. Abbildung 4.1 zeigt die Vorgehensweise zur Erstellung eines Hash-Werts.

$$M = BT \oplus R$$

$$C = Hash(M)$$

Eine weitere Möglichkeit zur Anonymisierung stellt der in Abschnitt 2.3.3 vorgestellte Algorithmus zur Generierung eines MAC (HMAC) dar. Dieses Verfahren wird in Abbilung

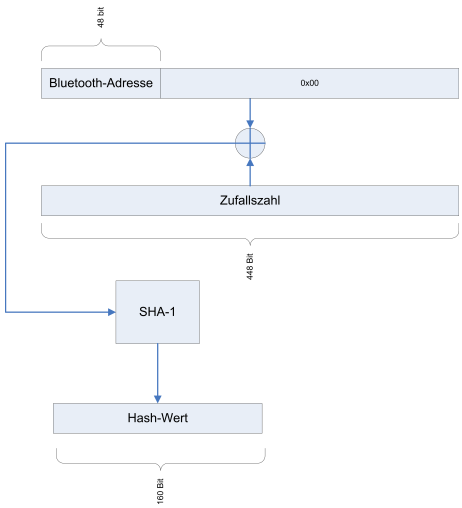

Abbildung 4.1: Anonymisierung

4.2 dargestellt. Bei HMAC (vgl. [2]) handelt es sich um ein etabliertes System mit untersuchter Sicherheit. Daher ist diesem Verfahren gegenüber dem einfachen Verfahren zur Anonymisierung der Vorzug zu geben. Nachteilig bei HMAC ist, dass zur Anonymisierung mehrere Rechnenschritte notwendig sind.

Bei dem in Abschnitt 3.3.3 vorgestellten Konzept erfolgt die Anonymisierung mit zwei verschiedenen Zufallszahlen. Daher entstehen in weiterer Folge auch zwei verschiedene Hash-Werte (C_1 und C_2, wobei C_1 jeweils mit der jüngeren Zufallszahl verschlüsselt wurde). Dieser Vorgang ist in Abbildung 4.3 erläutert.

4.1.2 Zeitlicher Erfassungsbereich

Der zeitliche Erfassungsbereich muss so gewählt werden, dass ein Großteil aller Fahrzeuge in einem Streckenabschnitt erfasst und nachverfolgt werden können. Der Erfassungsbereich kann über das Wechselintervall der Zufallszahlen eingestellt werden. So führt ein zu kurzes Wechselintervall zu wenig Erfassungen bei langsamen Fahrzeugen. Ein zu langes Wechselintervall garantiert zwar die Erfassung aller Fahrzeuge, bietet jedoch auch einen Angriffspunkt und ermöglicht eine Langzeitverfolgung von Fahrzeugen. Als guter Wert für das Wechselintervall wurde die Zeit ermittelt, in der 90% der Fahrzeuge den Streckenabschnitt durchqueren können. In Abbildung 4.4 wird die Problematik zwischen Erfassungsbereich und Wechselintervall dargestellt. Die Ermittlung dieses Zeitwerts kann theoretisch oder durch praktische Messungen erfolgen.

4.1.3 Verarbeitung der Daten am Datenbankserver

Der Datenbankserver muss die anoymisierten Werte einander zuweisen. Von den Sensorstationen erhält er Datenpakete, die zwei verschiedene anonymisierte Werte enthalten (C_1 und C_2). Anfangs wird der Jüngere der beiden Werte (C_1) mit den bereits gespeicherten Werten verglichen. Gibt es eine Übereinstimmung, kann sofort eine Berechnung der für die Verkehrsflußanalyse relevanten Daten durchgeführt werden. Wird der Wert jedoch nicht in der Datenbank aufgefunden, ist eine weitere Suche mit dem zweiten Wert (C_2) notwendig. Kommt es hier zu einem Treffer wird eine Verkehrsflussanalyse durchgeführt.

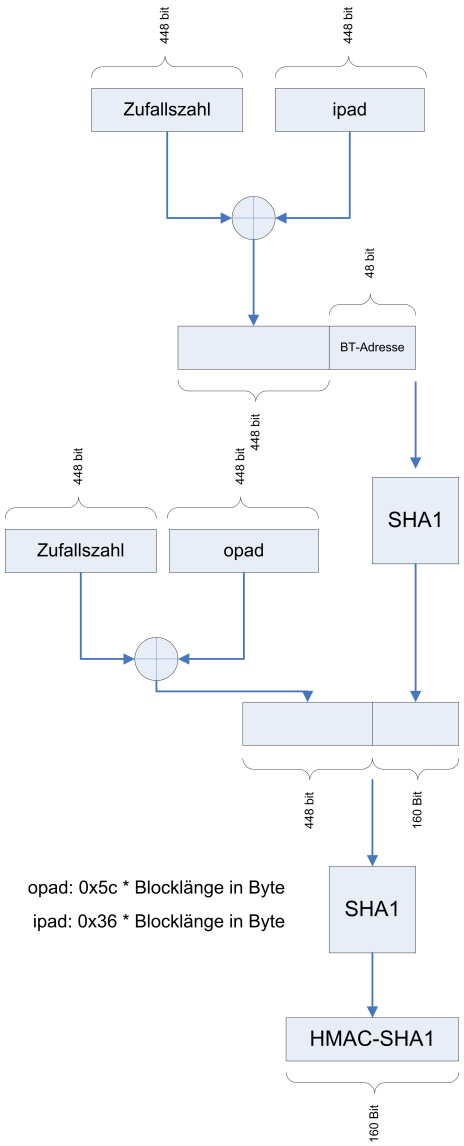

Abbildung 4.2: Anonymisierung mittels HMAC

Kann der Wert wieder nicht gefunden werden, ist noch keine Erfassung durchgeführt worden und der Wert erhält einen neuen Eintrag in der Datenbank. Bei jedem Treffer in der Datenbank wird der Hash-Wert mit dem aktuellsten Wert aus der Messung auf den neuesten Stand gebracht. Dieser Vorgang wird in Abbildung 4.5 dargestellt.

4.1.4 Public Key Infrastructure und Protokolle

Die Public Key Infrastructure hat die Aufgaben, die Kommunikationswege gegen Unbefungte zu schützen und eine gegenseitige Authentifizierung zu ermöglichen. Für diese beiden Aufgaben ist ein System aus asymmetrischen Schlüsselpaaren am besten geeignet. Mit Hilfe dieser Schlüsselpaare kann eine symmetrische Verschlüsselung initiiert werden, Zufallszahlen können ausgetauscht werden und die einzelnen Komponenten können sich gegenseitig authentifizieren. Zur Verwaltung der Schlüsselpaare ist eine zusätzliche In-

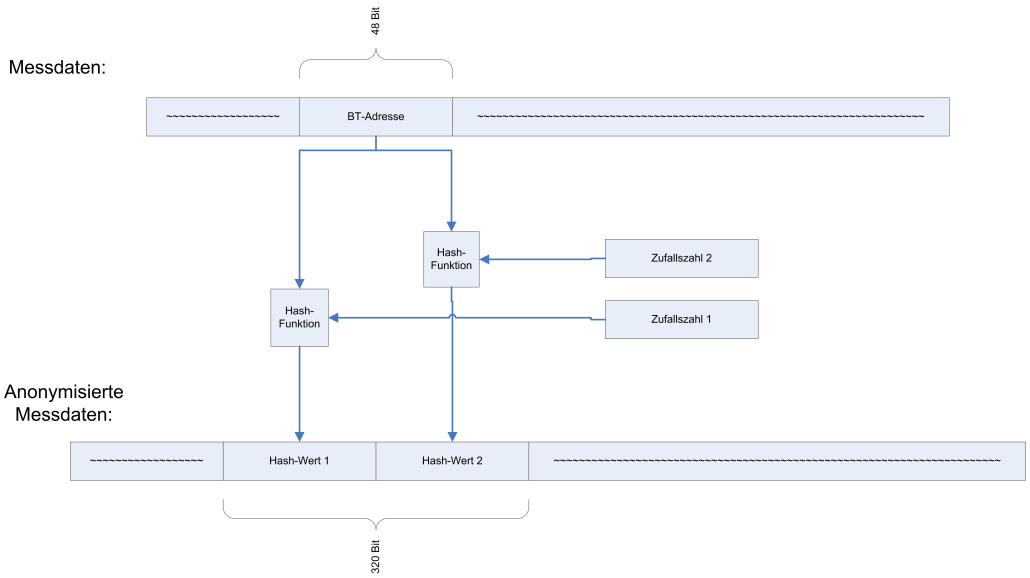

Abbildung 4.3: Verarbeitung der Messdaten

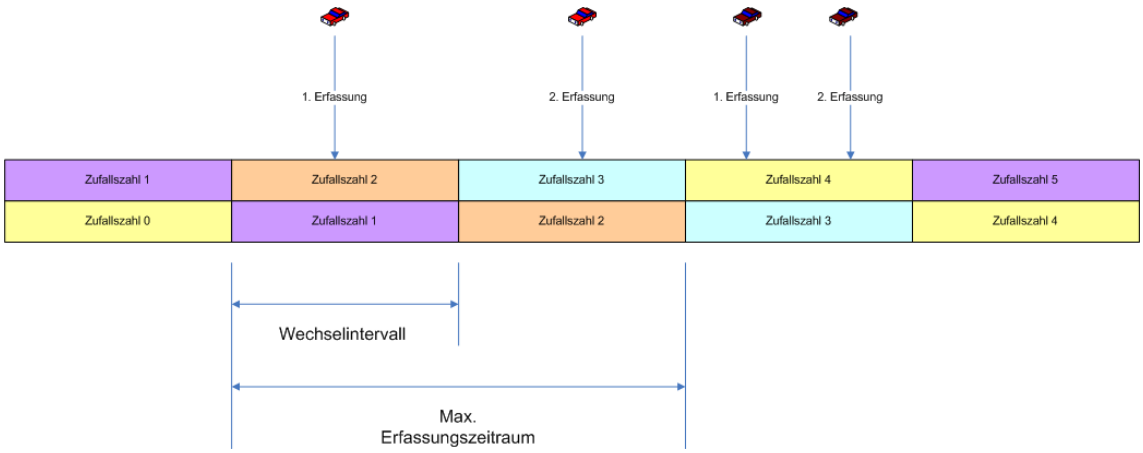

Abbildung 4.4: Erfassung und Wechselintervall bei der Anonymisierung

stanz vorgesehen (CA, Certificate Authority). Die CA übernimmt die in Kapitel 2.5.2 beschriebenen Aufgaben. Dazu wird zuerst von der CA ein selbstsigniertes Zertifikat mit dem öffentlichen Schlüssel der CA erstellt. Dieses Zertifikat muss vor Inbetriebnahme der anderen Komponenten auf diesem untergebracht werden. Als Alternative zum selbstsignierten Zertifikat kann auch ein Zertifikat für die CA bei einer Zertifizierungsstelle gekauft werden. Des Weiteren wird für jede Komponente ein Langzeitschlüsselpaar und ein zugehöriges Zertifikat erstellt. Auch diese müssen bei der Installation auf der Komponente gespeichert werden. Zertifikate entsprechen dem im Abschnitt 2.5.2 beschriebenen x509 Zertifikat.

Geht nun zum Beispiel eine Sensorstation in Betrieb, fordert sie bei der CA ein neues Schlüsselpaar an. Dieses wird, gesichert mit dem Langzeitschlüsselpaar, an die Sensorstation übertragen. Zusätzlich wird auch die aktuelle Certificate Revocation List (CRL) übertragen. Die Sensorstation speichert dieses Schlüsselpaar und überträgt das von der CA signierte Zertifikat zu diesem Schlüsselpaar an den Anonymisierungsserver. Dieser

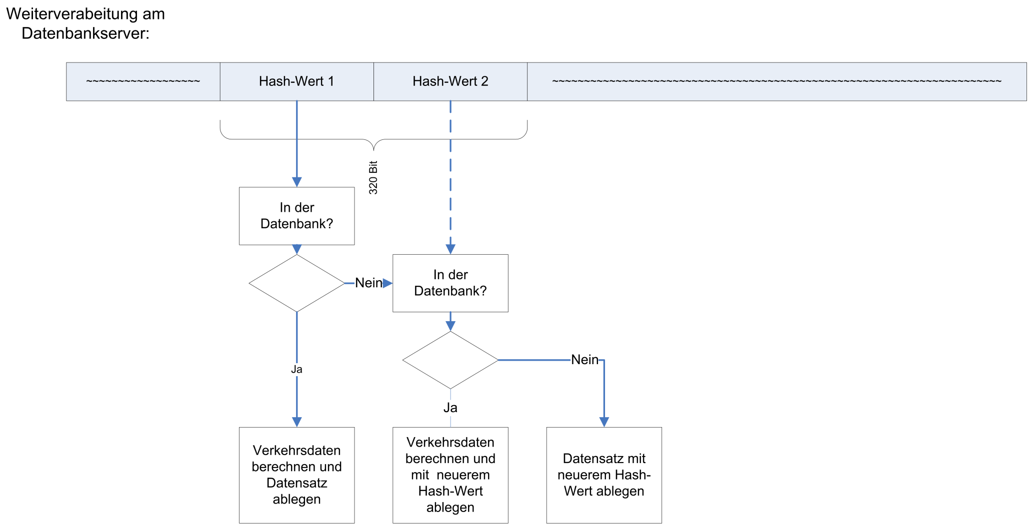

Abbildung 4.5: Verarbeitung der Daten am Datenbankserver

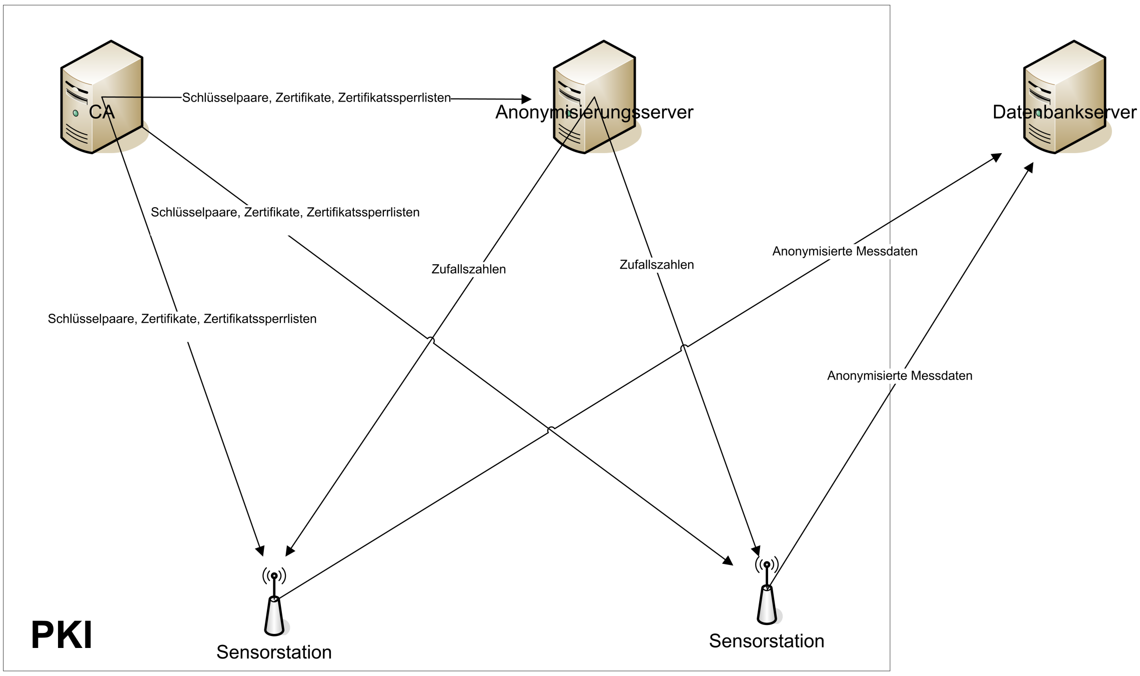

Abbildung 4.6: Mögliche Struktur mit PKI

überprüft das Zertifikat und ist somit über die Existenz und die Identität der Sensorstation informiert. Der Anonymisierungsserver nimmt die Sensorstation in die Liste für den Versand der Zufallszahlen auf. Zusätzlich überträgt der Anonymisierungsserver sein Zertifikat über das aktuelle Schlüsselpaar an die Sensorstation.

Ist nun ein Wechsel des aktuellen Schlüsselpaars notwendig, kann dies über das auf den Komponenten gespeicherte Langzeitschlüsselpaar erfolgen. Die CA verfügt über eine Liste der ausgegebenen Schlüsselpaare und deren Gültigkeitszeitraum. Bei Ablauf der Gültigkeit wird die CA tätig und überträgt ein neues Schlüsselpaar an die Komponente. Dieses Schlüsselpaar wird mit dem öffentlichen Langzeitschlüssel der Komponente gesichert. Die

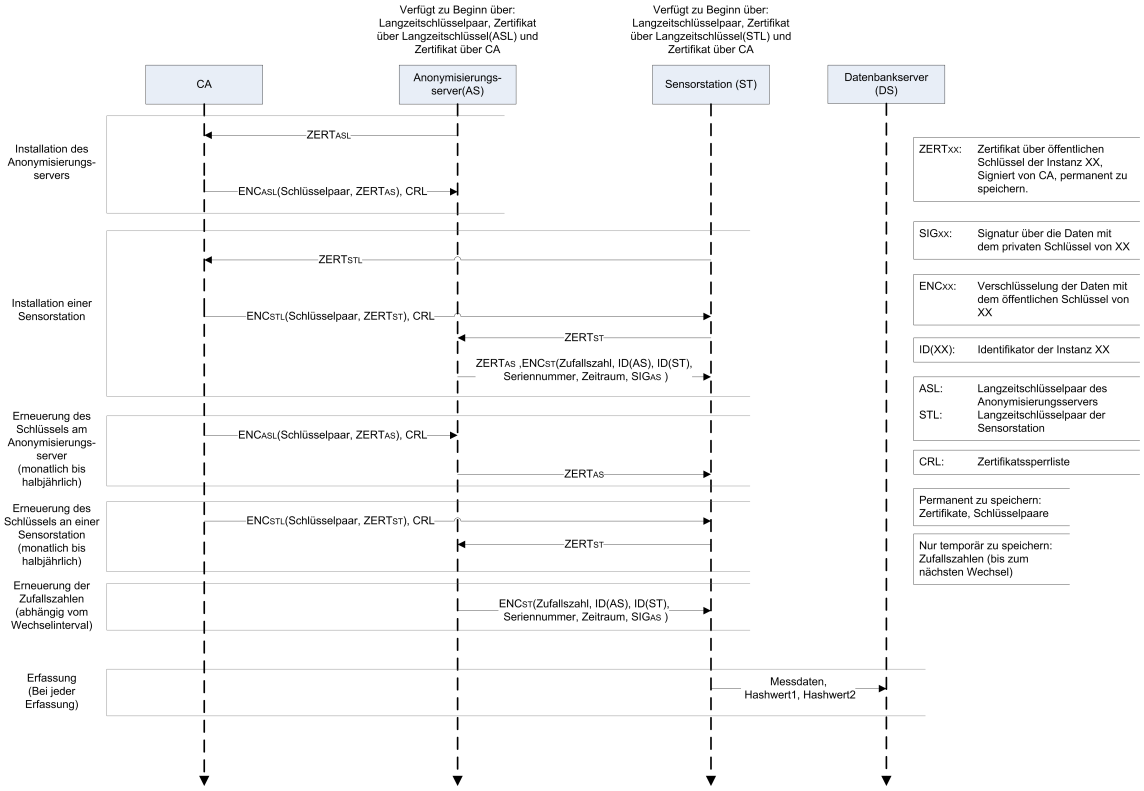

Abbildung 4.7: Sequenzdiagramm zum Betrieb der PKI

Komponente selbst sendet das Zertifikat über den neuen Schlüssel an alle anderen Komponenten, mit denen eine Kommunikation notwendig ist.

Eine Erneuerung der Zufallszahlen erfolgt im Wechselintervall. Dazu wird die Zufallszahl in eine ähnliche Struktur wie ein x509 Zertifikat verpackt. Diese Struktur muss die Zufallszahl, eine Seriennummer, einen Gültigkeitszeitraum und die ID des Anonymisierungsservers enthalten. Das gesamte Paket wird mit dem öffentlichen Schlüssel der Sensorstation gesichert und an diese übertragen.

4.2 Pseudonymisierung

Bei der Pseudonymisierung wird jeder Bluetooth-Adresse ein eindeutiger Identifikator zugewiesen. Dies wird von einer zusätzlichen Komponente erledigt (Pseudonymisierungsserver). Der Pseudonymisierungsserver erhält die Bluetooth-Adressen von den Sensorstationen und generiert daraus einen eindeutigen Identifikator. Näheres zum grundlegenden Konzept ist in Abschnitt 3.2 und 3.3.4 zu finden. Die Pseudonymisierung kann durch eine Hash-Funktion unter Einbeziehung eines Schlüssels erfolgen. Eine Alternative dazu wäre das Generieren einer Tabelle, in welcher zu jeder Bluetooth Adresse ein Identifikator abgelegt ist. Durch die Anzahl der möglichen Adressen (2^{48}) gilt diese Methode als unpraktisch. Eine Pseudonymisierung mit Hash-Funktion kann durch die in Abschnitt 2.3.3 dargestellte Methode (HMAC) erfolgen.

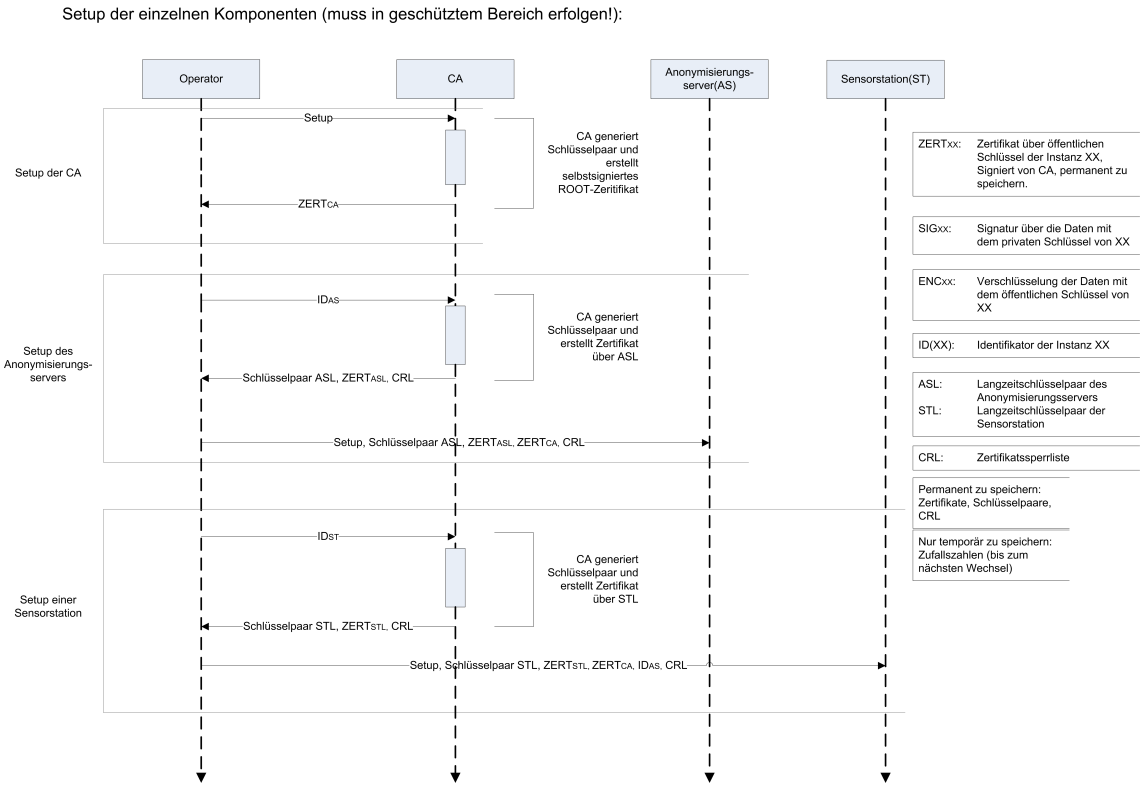

Abbildung 4.8: Sequenzdiagramm zum Setup der PKI

4.2.1 Schlüsselgenerierung und Schlüsselwechsel

Da der Schlüssel nur dem Pseudonymisierungsserver bekannt ist und der Weg der Pseudonymisierung daher auch nur von diesem Server wiederholt werden kann, kann ein längerer Zeitraum zwischen dem Schlüsselwechsel liegen. Der Zeitraum muss so konzipiert sein, dass bei einem möglichen Bekanntwerden des Schlüssels keine Langzeitverfolgung möglich ist und eine möglichst geringe Datenmenge betroffen ist. Der Schlüsselwechsel sollte jedoch so gewählt werden, dass dieser nicht immer an den selben Tagen oder Tageszeiten durchgeführt wird (Schlüsselwechsel zum Beispiel alle 3 Tage und 23 Stunden, das kleinste gemeinsame Vielfache von 24 und $3 \cdot 24 + 23 = 95$ ist $24 \cdot 95$, der Schlüsselwechsel erfolgt daher alle 95 Tage zur selben Uhrzeit). Bei einem Schlüsselwechsel kommt es zu einem kurzen Zeitraum in dem nur eine minimale Verkehrsflusserfassung möglich ist. Würde dieser Vorgang täglich zum selben Zeitpunkt erfolgen, wäre es nicht möglich eine Aussage über die Verkehrsflüsse zu diesem Zeitpunkt zu treffen. Durch den Wechsel zu verschiedenen Zeitpunkten können diese Daten jedoch aus den restlichen Messtagen interpoliert werden.

Die Generierung eines neuen Schlüssels kann durch einen Pseudo Random Number Generator (PRNG) erfolgen.

Die Pseudonyme werden über einen sicheren Kanal an die Sensorstationen übertragen und von diesem anstatt der Adresse mit dem Datensatz an den Datenbankserver übertragen.

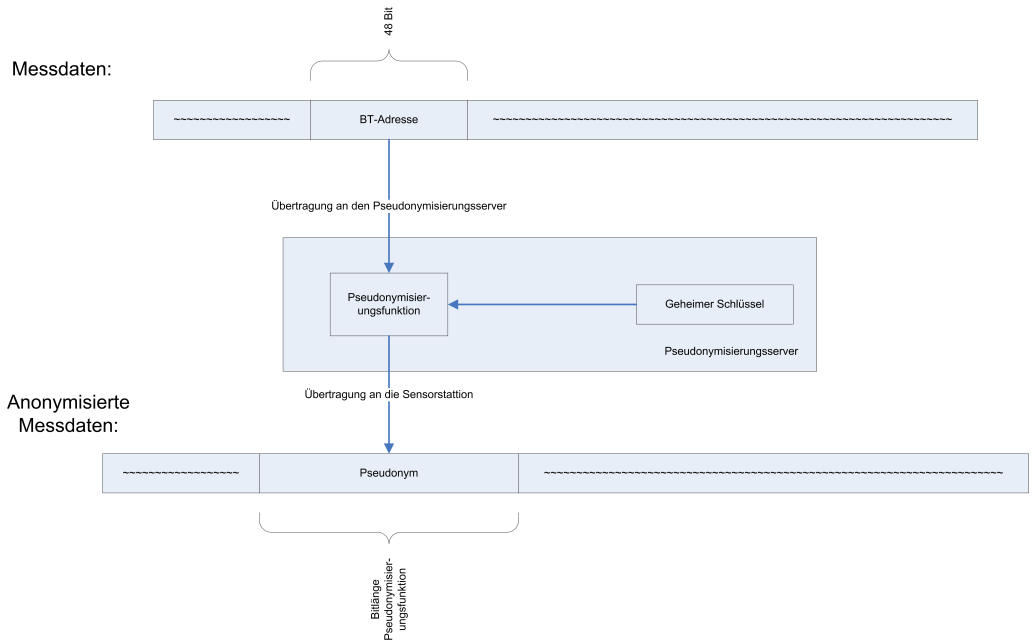

Abbildung 4.9: Austausch der Adresse durch das Pseudonym

4.2.2 Verarbeitung der Daten am Datenbankserver

Der Datenbankserver erhält den Datensatz mit Pseudonym statt Bluetooth-Adresse. Diese Informationen werden in die Datenbank eingetragen. Eine weitere Verkehrsflussanalyse kann nun wie mit der Bluetooth-Adresse umgesetzt werden.

4.2.3 TLS / Public Key Infrastructure

Zur Absicherung der Übertragung von Bluetooth-Adresse an Pseudonymisierungsserver und der Pseudonyme vom Server an die Sensorstationen kann eine TLS (siehe Abschnitt 2.5.1) zum Einsatz kommen. Zusätzlich wird dadurch eine Public Key Infrastructure, welche das nötige Schlüsselmanagement übernimmt, notwendig. Eine solche PKI kann ähnlich wie in Abschnitt 4.1.4 realisiert werden. Bei der Installation der Komponenten werden die Zertifikate ausgetauscht. Für den Pseudonymisierungsvorgang werden außer der Bluetooth-Adresse auch noch die Identifikatoren beider Komponenten sowie eine fortlaufende Seriennummer übertragen. Die Seriennummer wird für jeden Pseudonymisierungsvorgang von der Sensorstation erhöht. Der Pseudonymisierungsserver überprüft die Seriennummer (Sie muss höher sein als die zuletzt verwendete) und antwortet mit einem Paket mit der selben Seriennummer, dem Pseudonym und beiden Identifikatoren (siehe Abbildung 4.10)

Ein Austausch des Schlüsselpaars einer Komponente kann wie in Abschnitt 4.1.4 erläutert umgesetzt werden.

4.3 Implementierung auf den einzelnen Komponenten

Zu Testzwecken wurden Teile des Konzepts "Anonymisierung durch die Sensorstationen" (siehe Abschnitt 4.1) implementiert. Eine Fertigstellung des Projekts wird von der Firma

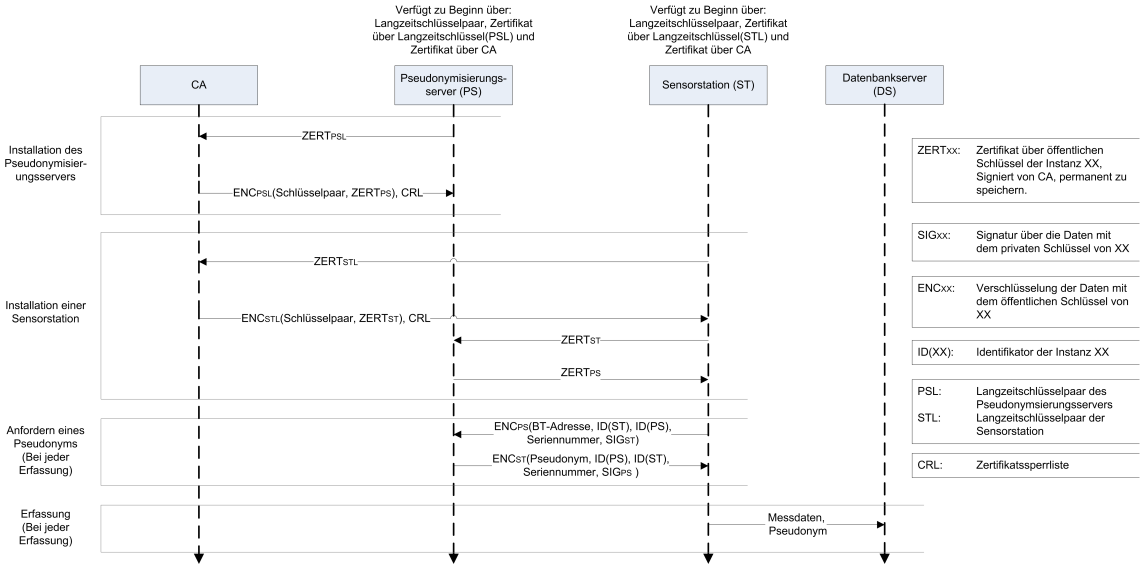

Abbildung 4.10: Sequenzdiagramm zum Betrieb der PKI bei Pseudonymisierung

C.C.COM erst bei Auftragserteilung angestrebt. Somit exisitieren Teile dieses Systems nur in einem Laborumfeld. Die Umsetzung gliedert sich in die verschiedenen Komponenten und die dort durchzuführenden Aufgaben.

4.3.1 Sensorstation

Auf der Sensorstation sind drei Erweiterungen nötig. Diese werden in Abbildung 4.11 dargestellt. Zum einem muss die Bluetooth-Adresse anonymisiert werden. Zum anderen ist die Verwaltung der Zertifikate und der Aufbau des sicheren Kanals zu Anonymisierungsserver und CA umzusetzen.

Als wichtigste Aufgabe führt die Sensorstation die Anonymisierung der Verkehrsdaten durch. Dies wurde analog zur Abbildung 4.1 und Abbildung 4.3 in Java implementiert. Für die kryptographischen Komponenten kommt die Java Cryptography Architecture (JCA) zum Einsatz. Die JCA stellt dabei den SHA1 Algorithmus zur Verfügung. In Java wurde eine Klasse BluetoothAnonymizer (siehe Abbildung) angelegt. Diese Klasse gliedert sich in die in Abbildung dargestellten Methoden.

stringXOR: Diese Methode verknüpft die MAC-Adresse und die Zufallszahl per XOR. Dabei wird auch die Länge der Zufallszahl(448 Bit) und der MAC-Adresse (48 Bit) überprüft. Falls diese Werte nicht den Vorgaben entsprechen, wird mit einer Fehlermeldung abgebrochen.

readFileAsString: Liest die Zufallszahlen aus den vorgegebenen Dateien.

stringSHA1: Führt die Hash-Funktion aus. Die Funktion selbst wird von der Java Crytography Architecture zur Verfügung gestellt. Als Eingangswert dient die mit der Zufallszahl verknüpfte MAC-Adresse.

toHexString: Konvertiert ein Byte-Bufferwert in einen String. Diese Methode wird benötigt um das Ergebnis der Hash-Funktion wieder in einen String umzuwandeln.

doAnonymization: Diese Methode führt die anderen Funktionen in der gegebenen Reihenfolge aus. Zuerst werden zwei Zufallszahlen aus den Dateien gelesen. Diese werden mit der gegebenen MAC-Adresse verknüft. Aus beiden Werten wird je ein Hash-Wert

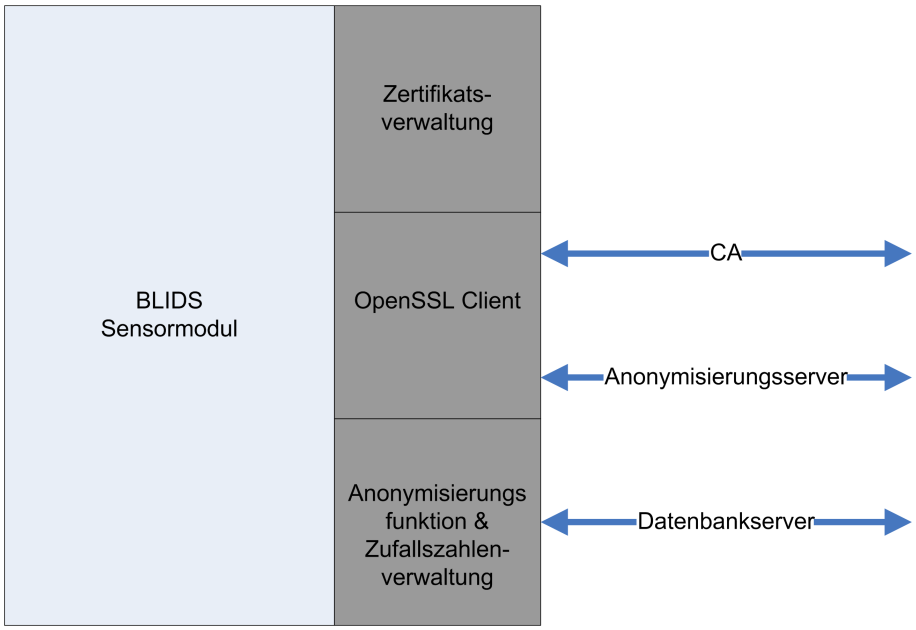

Abbildung 4.11: Erweiterungen auf der Sensorstation

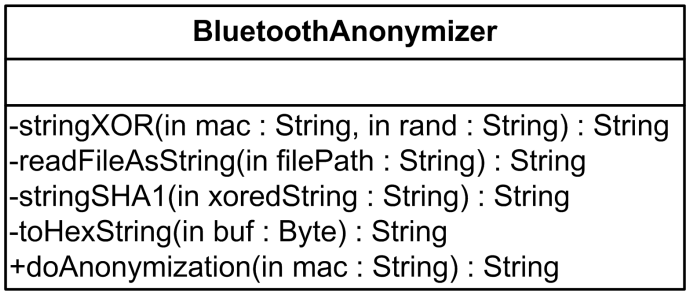

Abbildung 4.12: Klasse BluetoothAnonymizer

errechnet und dieser ausgegeben.

Weiters wurde das freie Packet OpenSSL[1] für die Verwaltung der Zertifikate und Sicherung des Kanals zum Anonymisierungsserver eingesetzt. Die Sensorstation wurde dabei als Client analog zu folgendem Tutorial[2] implementiert. Die Grundfunktion beinhaltet den Aufbau eines sicheren Kanals. Dazu wird zuerst mittels init_CTX der SSL Kontext aufgebaut. Weiters wird die TCP Verbindung hergestellt und aufbauend auf diese Verbindung der SSL Kanal hergestellt. Eine Weitere Funktion sorgt dafür das Daten an den Server übertragen oder von dort gelesen werden können.

4.3.2 Datenbankserver

Am Datenbankserver wurden keine Änderungen durchgeführt. Die Auswertung der erhaltenen Daten wird weiterhin von der Firma C.C.COM auf Wunsch des Kunden implementiert.

[1] http://www.openssl.org
[2] http://www.ibm.com/developerworks/linux/library/l-openssl2.html

4.3.3 Anonymisierungsserver

Am Anonymisierungsserver wurden zwei Teilbereiche implementiert. Die Generierung der Zufallszahlen erfolgt mittels Java.

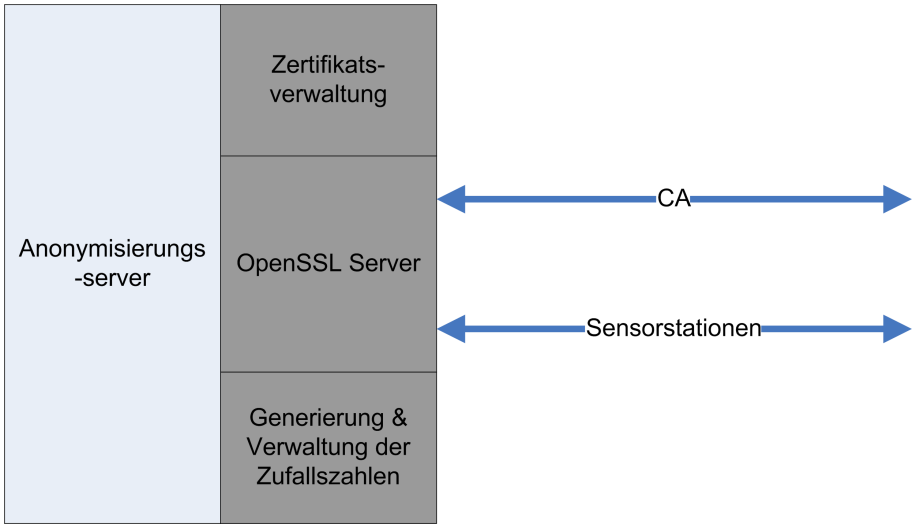

Abbildung 4.13: Umsetzungen auf dem Anonymisierungsserver

Die generierten Zufallszahlen werden als Dateien in einem Ordner abgelegt:

```
openssl req -config cccom_random.config -new -x509
-key cccom_random1.pem -out cccom_random1.csr -days 30
```

Weiters wird der Aufbau des sicheren Kanals und die Verwaltung der Sensorstationen mittels OpenSSL umgesetzt. Dazu werden alle Zertifikate von Sensorstationen in einem Ordner abgelegt. Diese Zertifikate werden von der CA laut Diagramm 4.7 übertragen. Die Implementierung der SSL Komponenten, basierend auf einem Tutorial[3], hat die Aufgabe einen OpenSSL Server zur Verfügung zu stellen, der bei Bedarf die Dateien mit den Zufallszahlen an die Sensorstation überträgt. Dies wurde zu Testzwecken nur rudimentär implementiert.

4.3.4 Certificate Authority

Für diese Komponente kommt wieder das OpenSSL Paket zum Einsatz. Folgendes Tutorial [4] diente als Hilfe bei der Erstellung der Zertifikate. Die CA erstellt zuerst ein selbstsigniertes Zertifikat:

```
openssl genrsa -aes256 -out private/cccom-cakey.pem 2048
openssl req -new -x509 -days 3650
-key private/cccom-cakey.pem -out cccom-ca.pem -set_serial 1
```

Dieses Zertifikat muss händisch auf allen weiteren Komponenten (Sensorstationen, Anonymisierungsserver) untergebracht werden. Weiters wird für den Anonymisierungsserver und für jede Sensorstation ein Zertifikat für den Setupprozess generiert. Dazu wird zuerst ein Zertifikatsanfrage (Certificate Signing Request - CSR) erstellt:

[3] http://www.cs.odu.edu/čs772/fall08/lectures/ssl_programming.pdf
[4] http://www.online-tutorials.net/security/openssl-tutorial/tutorials-t-69-207.html#schlsselpaar-fr-ca

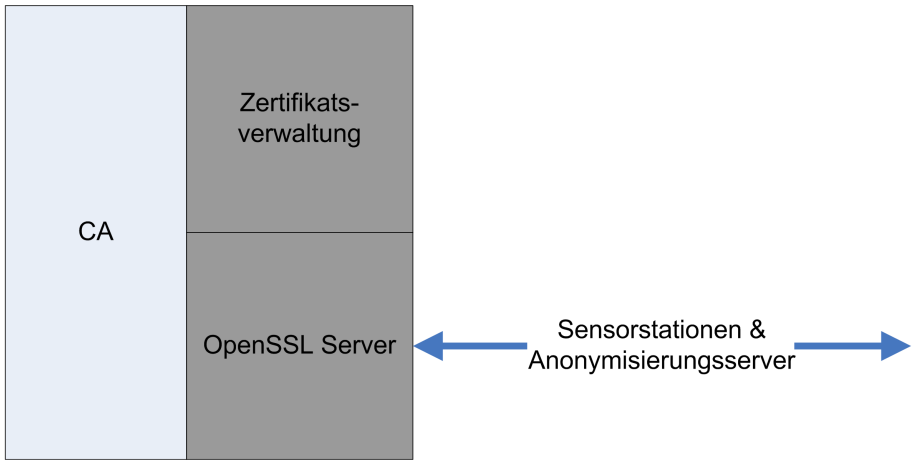

Abbildung 4.14: Umsetzungen auf dem Anonymisierungsserver

```
openssl req -new -newkey rsa:2048 -out
certs/anonymcsr.pem -nodes
-keyout private/anonymkey.pem -days 3650
```

Und mit Hilfe dieser Zertifikatsanfrage ein Zertifikat für jede Komponente erstellt (am Beispiel des Anonymisierungsservers):

```
openssl x509 -req -in
certs/anonymcsr.pem -out certs/anonymcert.pem -CA cccom-ca.pem -CAkey
private/cccom-cakey.pem -CAserial /usr/local/ssl/serial -days 3650
```

Diese werden ebenfalls manuell auf den Komponenten untergebracht. Diese Zertifikate und Schlüssepaare sind die Langzeitschlüssel, welche die Komponenten mit dem Anonymisierungsserver teilen. In weiterer Folge wird für jede Komponente ein temporäres Schlüsselpaar und Zertifikat erstellt. Diese dienen zur Kommunikation von Anonymisierungsserver und Sensorstationen und werden in regelmäßigem Abstand erneuert. Die Zertifikate werden dazu von der CA neu ausgestellt und auf eine Anfrage der Komponente an diese ausgeliefert. Dieser Teil wurde bisweilen nicht implementiert.

Kapitel 5

Schlussfolgerungen, Ausblick

Wie in den vorigen Kapiteln gezeigt, existieren durchaus Möglichkeiten, wie man Verkehrsflusserfassung umsetzen kann, ohne mit dem Datenschutz in Konflikt zu geraten. Das Datenschutzgesetz ist in diesem Bereich sehr schwammig formuliert. Daher wird vom Gesetzgeber in Zukunft eine genauere Definition für den Bereich Verkehrsflusserfassung notwendig sein. So ist noch unklar, ob die Bluetooth-Adresse zu den persönlichen Daten gehört oder wie lange Verkehrsdaten nachvollziehbar sein dürfen. Klar ist, dass solche Informationen ein genaues Bewegungsbild einer Person skizzieren können. Dazu bedarf es nur der Verknüpfung verschiedener Datenbanken. Die in dieser Diplomarbeit vorgestellten Konzepte greifen einer solchen Überarbeitung der Gesetzestexte bereits vor. So genügt das in Abschnitt 4.1 ausgearbeitete Konzept möglichen Anforderungen, die in Zukunft an das System gestellt werden könnten. Dabei werden Daten nur temporär verknüpfbar angelegt. Weiters wird durch einen zusätzlichen Server verhindert, dass von einer gegebenen Bluetooth-Adresse auf die Einträge in der Datenbank geschlossen werden kann. Symmetrische und asymmetrische Algorithmen sorgen in Form von TLS für die Absicherung der Kanäle zwischen den Komponenten. Das so umgesetze System bietet die Sicherheit, dass durch die Übernahme einer einzelnen Komponente durch einen Angreifer, diesem kein weiterer Nutzen entsteht. Die genutzten kryptographischen Komponenten entsprechen dem Stand der Technik und bieten auch für die nähere Zukunft ausreichend Sicherheit. Jedoch muss bei Änderungen der Gesetzestexte nochmals überprüft werden, ob das System den Anforderungen entspricht oder das Benötigte nicht vielleicht auch mit geringeren Mitteln umsetzbar wäre. Das besprochene Konzept liegt bereits teilweise implementiert vor und wird vermutlich in Zukunft in die Systeme der Firma C.C.COM integriert.

Kapitel 6

Zusammenfassung

Ziel dieser Diplomarbeit war der Entwurf eines Systems um Verkehrsflusserfassung möglichst den Kriterien des Datenschutzes entsprechend umzusetzen. Dazu wurde in Kapitel 1 das von der Firma C.C.COM entwickelte System zur Verkehrserflußerfassung vorgestellt. Weiters wurden in diesem Kapitel die relevanten Technologien, wie Bluetooth, erklärt. Ein kurzer Abschnitt über die gesetzlichen Definitionen des Datenschutz schließen das Kapitel ab.

In Kapitel 2 wurden alle kryptographischen Algorithmen, die für eine Umsetzung relevant sind, erklärt und gegenübergestellt. Dies umfasst symmetrische Verschlüsselungsalgorithmen wie AES, symmetrische Algorithmen wie RSA, Hash-Funktionen, die Erzeugung von Zufallszahlen, Protokolle und Zertifikate.

Aus diesen Einzelkomponenten wurden in weiterer Folge Konzept entwickelt. Diese Konzepte werden in Kapitel 3 erläutert. Zu Beginn dieses Kapitels wird nochmals in Kürze die Problemstellung erklärt. Für zwei dieser Konzepte wird im darauf folgenden Kapitel (Kapitel 4) eine konkrete Umsetzung definiert. Die beiden Konzepte wurden in Abstimmung mit der Firma C.C.COM anhand ihrer Vorzüge ausgewählt. Eines dieser Konzepte wurde weiter ausgearbeitet und in einer Testumgebung implementiert.

Anhang A

Definitionen

A.1 Abkürzungen

AES	Advanced Encryption Standard
DES	Data Encryption Standard
CA	Certificate Authority
CRL	Certificate Revocation List
CSR	Certificate Signing Request
EDGE	Enhanced Data Rates for GSM Evolution
EM	Encoded Message
EMSA	Encoded Methode for Signatures with Appendix
FHS	Frequency Hopping Synchronization
GPRS	General Packet Radio Service
I2OSP	Integer to Octet String Primitive
IDP	Identity Provider
ISM	Industrial, Scientific and Medical
IV	Initial Vector
MDC	Message Detection Code
MGF	Mask Generation Function
OS2IP	Octet String to Integer Primitive
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
PRIME	Privacy and Identity Management for Europe
PRNG	Pseudo Random Number Generator
PS	Padding Sequenze
PSS	Probabilistic Signature Scheme
QoS	Quality of Service
RFID	Radio Frequency Identification
RP	Relying Party
RSA	Rivest Shamir Adleman
RSAVP	RSA Verification Primitive
RSASP	RSA Signature Primitive
SIM	Subscriber Identity Module
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security

TPM	Trusted Platform Module
WLAN	Wireless Local Area Network

Literaturverzeichnis

[1] S. Aktiengesellschaft. Patentschrift, verfahren zur ermittlung einer reisezeit. `http://www.patent-de.com/20091217/DE102008028112A1.html`.

[2] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication, 1996.

[3] A. Beutelspacher. *Kryptologie*. Vieweg Verlagsgesellschaft, 2005.

[4] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, and A. Shamir. Key recovery attacks of practical complexity on aes variants with up to 10 rounds. Cryptology ePrint Archive, Report 2009/374, 2009.

[5] A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. Cryptology ePrint Archive, Report 2009/317, 2009.

[6] A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. In *ASIACRYPT*, pages 1–18, 2009.

[7] A. Biryukov, D. Khovratovich, and I. Nikolic. Distinguisher and related-key attack on the full aes-256. In *CRYPTO*, pages 231–249, 2009.

[8] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[9] I. u. T. Bundesministerium für Verkehr. Straßenverkehrsordnung. `http://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10011336`, 1960.

[10] C. Cannière and C. Rechberger. Preimages for reduced sha-0 and sha-1. In *CRYPTO 2008: Proceedings of the 28th Annual conference on Cryptology*, pages 179–202, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] C. D. Canni‘ere, F. Mendel, and C. Rechberger. Collisions for 70-step sha-1: On the full cost of collision search. In C. M. Adams, A. Miri, and M. J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *LNCS*, pages 56–73. Springer, 2007.

[12] C. D. Canni‘ere and C. Rechberger. Finding sha-1 characteristics: General results and applications. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.

[13] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[14] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[15] W. Commons. Cryptography diagramms — wikimedia commons, 2009. [Online; accessed 2-January-2010].

[16] J. Daemen and V. Rijmen. The rijndael block cipher: Aes proposal, 1999.

[17] J. Daemen and V. Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[18] T. Dierks and E. Rescorla. The transport layer security (tls) protocol, version 1.2. `http://tools.ietf.org/rfcmarkup/5246`, 2008.

[19] M. Dworkin. Recommendation for block cipher modes of operation. NIST Special Publication 800-38A, 2004.

[20] J. K. Elaine Barker. Recommendation for random number generation using deterministic random bit generators. NIST Special Publication 800-90, 2007.

[21] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of rc4. In *RC4, Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography*, pages 1–24, 2001.

[22] B. für Justiz. Bundesdatenschutzgesetz. `http://bundesrecht.juris.de/bdsg_1990/__3.html`, 2009.

[23] D. Z. für Luft-und Raumfahrt. Verkehrserfassung. `http://www.dlr.de/desktopdefault.aspx/tabid-590/962_read-1231/`.

[24] A. für Verkehrsmanagement Landeshauptstadt Düsseldorf. Detektionssysteme. In *Moderne Technik zur Verkehrserfassung*, 2003.

[25] S. Horman. Ssl and tls, an overview of a secure communications protocol. `http://horms.net/projects/ssl_and_tls/stuff/ssl_and_tls.pdf`, 2005.

[26] B. S. Inc. Specifiation of the bluetooth system. `http://www.bluetooth.com/NR/rdonlyres/F8E8276A-3898-4EC6-B7DA-E5535258B056/6545/Core_V21__EDR.zip`, 2007. [Online; Februar, 2008].

[27] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[28] r. Laboratories. *PKCS #1 v2.1: RSA Cryptography Standard*, Apr 2002.

[29] R. Labs. Pkcs#1 v2.1: Rsa cryptography standard, 2001.

[30] T. H. Martin Fellendorf. 4d verkehrsmodelle: Verkehrsplanerische lösungen entwickeln und vermitteln. `http://www.corp.at/corp_relaunch/papers_txt_suche/CORP2003_Haupt.pdf`.

[31] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996. [Online; Februar, 2008].

[32] NIST. Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication 197, 2001.

[33] NIST. Secure hash standard (shs). Federal Information Processing Standards Publication 180-3, 2008.

[34] NIST. Digital signature standard (dss). Federal Information Processing Standards Publication 186-3, 2009.

[35] D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: The case of aes, 2006.

[36] E. Oswald. Lecture notes: It security. `http://www.iaik.tugraz.at/content/teaching/master_courses/it_sicherheit/downloads/`. [Online; April, 2008].

[37] M. E. Oswald. *Einsatz und Bedeutung elliptischer Kurven für die elektronische Signatur*. IAIK, 2001.

[38] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Proxies for anonymous routing. In *ACSAC '96: Proceedings of the 12th Annual Computer Security Applications Conference*, page 95, Washington, DC, USA, 1996. IEEE Computer Society.

[39] V. Rijmen and E. Oswald. Update on sha-1. Cryptology ePrint Archive, Report 2005/010, 2005. `http://eprint.iacr.org/`.

[40] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, 1983.

[41] D. Saha, D. Mukhopadhyay, and D. RoyChowdhury. A diagonal fault attack on the advanced encryption standard. Cryptology ePrint Archive, Report 2009/581, 2009.

[42] Y. Sasaki, L. Wang, and K. Aoki. Preimage attacks on 41-step sha-256 and 46-step sha-512. Cryptology ePrint Archive, Report 2009/479, 2009. `http://eprint.iacr.org/`.

[43] M. Sauter. *Grundkurs Mobile Kommunikationssysteme*. Vieweg, 2006.

[44] N. Smart. *Cryptography: An Introduction*. Mcgraw-Hill Professional, 2002. [Online; Oktober, 2009].

[45] I. E. E. E. C. Society. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wirelees Personal Area Networks (WPANs)*. IEEE Computer Society, ieee std 802.15.4 2006 edition, 2006.

[46] W. Stallings. *Cryptography and Network Security*. Prentice Hall, 1998.

[47] D. R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall, 2006.

[48] A.-K. A. Tamimi. Performance analysis of data encryption algorithms. `http://www.cs.wustl.edu/~jain/cse567-06/ftp/encryption_perf/index.html#2_2_1`, 2006. [Online; Februar, 2008].

[49] M. Ullmann, F. Koob, and H. Kelter. Anonyme online-wahlen - lösungsansätze für die realisierung von online-wahlen. *Datenschutz und Datensicherheit*, 25(11), 2001.

[50] E. Union. Richtlinie zur vorratsdatenspeicherung, 2006.

[51] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. In *In Proceedings of Crypto*, pages 17–36. Springer, 2005.

[52] J. Weinzerl. *BLIDS Sensor: Embedded System - zur Verkehrserhebung.* C.C.COM, 2009.

[53] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and privacy aspects of low-cost radio frequency identification systems, 2004.

[54] P. Wohlmacher. *Digitale Signaturen und Sicherheitsinfrastrukturen.* IT Verlag, 2001.