Diplomarbeit

PRAKTISCHER EINSATZ VON ZERO KNOWLEDGE-VERFAHREN ZUR AUTHENTIFIZIERUNG

ausgeführt zum Zweck der Erlangung des akademischen Grades einer/eines
„Diplom-Ingenieurin/Diplom-Ingenieurs für technisch-wissenschaftliche Berufe“
am Masterstudiengang Telekommunikation und Medien
der Fachhochschule St. Pölten

unter der Erstbetreuung von

Univ.-Doz. Dipl.-Ing. Dr. Ernst Piller

Zweitbegutachtung von

FH-Prof. Dipl.-Ing. Johann Haag

ausgeführt von

Markus Brandl, BSc
tm0710262004

Ort, Datum Unterschrift

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der von den Begutachtern beurteilten Arbeit überein.

Ort, Datum *Unterschrift*

Kurzfassung

Diese Arbeit beschäftigt sich mit dem Thema *Zero Knowledge*. Diese Technologie ist eine Methodik zur Authentifizierung bzw. Identifizierung einer Entität. Die spezielle Eigenschaft, die Zero Knowledge beschreibt, ist, dass bei der Durchführung eines Zero Knowledge-Protokolls keine Informationen über das Geheimnis öffentlich werden und missbraucht werden können. Da oftmals Challenge-Response-Techniken eingesetzt werden, wird in dieser Arbeit analysiert, ob Zero Knowledge praktischen Einsatz finden kann und in welcher Weise dieser vonstatten gehen kann.

Mithilfe einer theoretischen Analyse der Thematik und Bewertung der Protokolle, soll die Praktikabilität von Zero Knowledge dargestellt werden.

In dieser Arbeit wird auch, neben weiteren Einsatzmöglichkeiten von Zero Knowledge, der Einsatz auf mobilen Geräten begutachtet, wobei eine Analyse der Übertragungsmedien Infrarot, Bluetooth und NFC durchgeführt und ein Protokoll in einfachster Weise implementiert wird.

Abstract

This theses deals with the issue of *Zero Knowledge*. This technology is a methodology for identification and authentication of an entity. The special characteristic of Zero Knowledge is, that within a run of a Zero Knowledge protocol no information about the secret becomes public and could be abused.

As often challenge-response techniques are used, this theses analyzes whether Zero Knowledge could find practical use and the manner in which this could take place.

Using a theoretical analysis of the issue and evaluating the protocols, the practicability of Zero Knowledge is presented. In this theses the use on mobile devices, among other uses of Zero Knowledge, will be examined too, with an analysis of the transmission medias infrared, Bluetooth and NFC, and a simple protocol is implemented.

Vorwort

Diese Diplomarbeit basiert auf der Grundlage der Lehrveranstaltung Advances Network Architectures bei Herrn Univ.-Doz. Dipl.-Ing. Dr. Ernst Piller. In dieser Lehrveranstaltung wurde das Themengebiet *Zero Knowledge* jedoch nur kurz und in theoretischer Form dargebracht. Da diese Thematik mein Interesse geweckt hat, soll dies in dieser Diplomarbeit vertieft werden.

Ziel dieser Diplomarbeit ist die Veranschaulichung des Themengebiets *Zero Knowledge* anhand praktischer Einsatzmöglichkeiten und der Entwicklung einer Java-Applikation, wobei ein Authentifizierungsvorgang zwischen zwei mobilen Devices durchgeführt wird.

Das Hauptaugenmerk dieser Arbeit liegt beim Einsatz dieser Authentifizierungsmethode auf mobilen Devices. Hierbei sollen die Übertragungsmedien Infrarot, Bluetooth und NFC betrachtet und auf ihre Einsatzfähigkeiten überprüft werden.

Jedoch wird auch der Einsatz dieser Methode mit Chipkarten erläutert, wobei dies nur kurz in theoretischer Form stattfinden wird.

Auch weitere allgemeine Überlegungen für den Einsatz von *Zero Knowledge* sollen gefunden werden.

Hinweis:

Diese Arbeit wird in einer gender-unabhängigen Form geschrieben, d.h. Begriffe wie z.B. „Benutzer“, „User“, etc. sind allgemein und nicht geschlechtsspezifisch zu verstehen.

Weiters werden in dieser Arbeit englische Fachbegriffe verwendet, da diese auch im deutschen Sprachraum üblich sind.

Inhaltsverzeichnis

1 Einleitung **1**

2 Zero Knowledge **3**

2.1 Probleme durch unsichere Authentifizierungsmechanismen 4

2.2 Historische Entwicklung 4

2.3 Funktionsweise 5

2.3.1 Mathematische/Interaktive Beweise 6

2.3.2 Komplexitätsklassen 6

2.3.3 Beispiel zur Erklärung von Zero Knowledge 7

2.3.4 Grundlegendes Protokoll 9

2.3.5 Beispiel für ZK (Graphentheorie) 10

2.3.5.1 Exkurs: Isomorphie von Graphen 10

2.3.6 Interaktive Beweissysteme 12

2.3.6.1 Interaktive Beweissysteme und die Komplexitätsklasse IP . . 13

2.3.7 Unterteilung von Zero Knowledge 14

2.3.8 Grundlegende Probleme 15

2.3.8.1 Composition 15

2.3.9 Parallele Ausführung 16

2.3.10 Nicht-interaktive Zero Knowledge-Beweise 17

2.3.11 Effizienz-Messungen 18

2.3.12 Zusammenfassung 20

2.4 Protokolle 20

2.4.1 Generelle Struktur 20

2.4.2 Protokoll 1 - Fiat-Shamir 21

2.4.2.1 Durchführung 21

2.4.3 Protokoll 2 - Feige-Fiat-Shamir 22

2.4.3.1 Durchführung 22

2.4.4 Protokoll 3 - Guillou-Quisquater 22

2.4.4.1 Durchführung 23

2.4.5 Protokoll 4 - Schnorr 23

2.4.5.1 Durchführung 23

2.4.6 Bewertung der Protokolle 24

2.4.6.1 Fiat-Shamir 24

2.4.6.2 Feige-Fiat-Shamir 24

2.4.6.3 Goillou-Quisquater 25

2.4.6.4 Schnorr 25

2.5 Challenge-Response 25

2.5.1 Hintergrund 26

2.5.2 Funktionsweise 27

2.5.3 Vergleich mit Zero Knowledge 27

2.6 Angreifbarkeit bzw. Sicherheitsprobleme von Zero Knowledge 29

3 Übertragungsmedien 32

3.1 Allgemeines 32

3.2 Infrarot 33

3.2.1 Spezifikationen 33

3.2.2 Funktionsweise 34

3.2.2.1 IrDA-Stack 34

3.2.3 Applikationsentwicklung 34

3.3 Bluetooth 35

3.3.1 Entwicklung 35

3.3.2 Spezifikationen/Klassen 36

3.3.2.1 Spezifikationen 36

3.3.2.2 Klassen 37

3.3.3 Funktionsweise 37

3.3.4 Sicherheitsmerkmale 41

3.3.5 Limitationen von Bluetooth 42

3.3.6 Applikationsentwicklung 43

3.4 NFC 43

3.4.1 Entwicklung 43

3.4.2 Standards 44

3.4.3 Funktionsweise 44

3.4.3.1 General Protocol flow 45

3.4.3.2 Angriffsmöglichkeiten (vgl. [Ern06, S. 4ff]) 46
3.4.3.3 Einsatz 47
3.4.4 Applikationsentwicklung 48
3.5 Bewertung der verschiedenen Übertragungsmedien 48
3.5.0.1 IrDA 50
3.5.0.2 Bluetooth 50
3.5.0.3 NFC 50
3.5.1 Fazit 51

4 Einsatzmöglichkeiten von Zero Knowledge 52

4.1 WLAN-Authentifizierung 52
4.2 Authentifizierung an einem Server 54
4.3 ZK als Authentifizierungsschicht innerhalb von Protokollen 55
4.4 Geschützte Inhalte 56
4.4.1 Offline 56
4.4.2 Online 56
4.5 Authentifizierung im Web 56
4.6 Dongles 57
4.7 Token 57
4.8 weitere Möglichkeiten 58
4.9 Praktikabilität von Zero Knowledge 59

5 Einfache Forschungsstellung 60

5.1 Bedingungen 60

5.1.1 Grundlagen für die Programmierung 61
5.1.1.1 J2ME 61
5.1.1.2 Bluetooth 63
5.1.2 Ablauf 63
5.1.2.1 Screens 67
5.2 Erwartetes Ergebnis 67
5.3 Durchführung 68
5.4 Ergebnis 69
5.5 Fazit 70

6 Zukünftige Entwicklungen 72

7 Conclusio 74

8 Appendix 76

8.1 Sourcecode des Beispiels 76
8.1.1 BtZKApp.java 76
8.1.2 ActionManager.java 77
8.2 Alle Screens des Ablaufs der Applikation 86

Literaturverzeichnis 91

Abbildungsverzeichnis 93

Tabellenverzeichnis 96

Sourcecode-Verzeichnis 97

Kapitel 1

Einleitung

Kryptographie an sich ist bereits sehr lange ein Teil unserer Gesellschaft. Sie hat Einflüsse in fast jeden Lebensbereich. Speziell im Bereich des privaten Schutzes hat sich die Kryptographie in den letzen 20 Jahren stark erweitert und ist in erheblichem Maße erforscht worden.

Die moderne Kryptographie dient im Allgemeinen zur Erhaltung und Bezeugung von

- Confidentiality (Vertraulichkeit)
- Authentication (Authentifizierung)
- Integrity (Integrität - Unversehrtheit)
- Nonrepudiation (Nichtzurückweisung - User hat Message unweigerlich abgeschickt)

Das Thema dieser Diplomarbeit ist im Gebiet der Authentifizierung anzusiedeln, da *Zero Knowledge* rein zur Authentifizierung bzw. Identifizierung eingesetzt wird.

Viele Systeme verwenden für die Authentifizierung ein Passwortsystem. Dieses gilt jedoch als unsicher, da es die Unsicherheitsquelle Mensch benötigt. Weiters gelten andere Authentifizierungsmethoden durch verschiedene Attacken als nicht mehr sicher. Daher soll alternativ die Authentifizierung mittels *Zero Knowledge* analysiert werden.

Hierbei ergeben sich folgende Forschungsfragen:

- Welche Vor-/Nachteile stellt Zero Knowledge im Vergleich zu klassischen Authentifizierungsmechanismen dar?
- Ist Zero Knowledge praktikabel und einfach zu implementieren?
- Welche Voraussetzungen sind für den Einsatz von Zero Knowledge vonnöten?
- Welche Einsatzgebiete kann Zero Knowledge abdecken?
- Welche Übertragungstechnologie ist für Zero Knowledge zu empfehlen?

Der logische Aufbau der Arbeit wird unterteilt in grundsätzlich 4 Teile.
Der erste Teil dient der theoretischen Beschreibung und der Grundlagen von *Zero Knowledge* (Abk. ZK), wobei diese nicht nur auf die Theorie, sondern auch auf den Praxiseinsatz bezogen werden soll.
Der zweite Teil dient der Beleuchtung der Übertragungsmedien, die bei mobilen Devices eingesetzt werden. Diese sollen betrachtet und objektiv auf ihre Einsatzfähigkeit beurteilt werden.
Im dritten Teil werden praktische Einsatzmöglichkeiten dieser Authentifizierungsmethode gefunden und beschrieben, wie einfach bzw. ob diese praktisch umgesetzt werden können.
Der vierte Teil der Arbeit besteht aus einem praktischen Beispiel, in dem eine Authentifizierung zwischen zwei mobilen Devices über Bluetooth erfolgt.

Es soll auch ein grundsätzlicher Vergleich zur Authentifizierung über ein Challenge-Response-Verfahren durchgeführt werden.
Spezielle Behandlung findet der Einsatz von *Zero Knowledge* bei mobilen Devices und evtl. bei Chipkarten, da diese einen hohen Wert an Aktualität an sich haben.
Es ist jedoch nicht Thema dieser Diplomarbeit, Zero Knowledge-Protokolle zu erneuern oder zu verbessern, sondern die aktuell applikablen Protokolle zu analysieren und zu bewerten bzw. einzusetzen und einen Einblick in die Forschungen und den aktuellen Stand betreffend Zero Knowledge zu geben.

Kapitel 2

Zero Knowledge

Zero Knowledge-Verfahren sind kryptographische Protokolle, die auf Interaktivität basieren. Interaktive Beweise können aufgrund ihrer Interaktivität mehr mathematische Behauptungen zeigen, als dies klassische Beweise können. Weiters ermöglichen diese es, ein fast perfektes Protokoll darstellen zu können.(vgl. [Alb06, S. 36])

Kryptographische Protokolle müssen verschiedene Anforderungen erfüllen, um als gutes Protokoll zu gelten:

- Durchführbarkeit
  Wenn sich die beteiligten Parteien an die Spezifikationen des Protokolls halten, wird das erwartete Ergebnis erreicht werden. Dieses kann im Anschluss daran auch verifiziert werden.
- Korrektheit
  Sollte jemand versuchen, die richtige Lösung zu erraten bzw. zu betrügen, soll die Wahrscheinlichkeit dies zu erreichen, verschwindend gering sein.

Bei Zero Knowledge-Beweisen gibt es zusätzlich dazu noch eine dritte Eigenschaft. Diese Eigenschaft bedeutet, dass der Beweis erbracht werden kann, ohne dass das Geheimnis preisgegeben wird bzw. werden muss.

2.1 Probleme durch unsichere Authentifizierungsmechanismen

Nach wie vor sind speziell auch in IT-Umgebungen Passwörter sehr gebräuchlich. Also die Kombination von Benutzerkennung und dem dazugehörigen Passwort. Diese Authentifizierung wird bei vielen Internetdiensten (FTP, E-Mail etc.), aber auch bei Geldautomaten usw. verwendet, wobei die Benutzerinformation z.B. auf der Chipkarte hinterlegt ist. Es wird jedoch hierbei mittlerweile auf eine verschlüsselte Übertragung zurückgegriffen, wobei aber z.B. Man-in-the-Middle-Attacken für Angreifer zielführend sein können.

Auch das Thema Biometrie spielt für die Authentifizierung eine wichtige Rolle. Diese Authentifizierungsmethode ist zwar weitgehend akzeptiert und für die Authentifizierung eingesetzt, hat aber ein anderes Ziel als Zero Knowledge und ist somit nicht direkt damit vergleichbar.[1]

Bei Kryptographie gibt es auch weitere Formen von Attacken, die auf bestimmte Protokolle anwendbar sind. In [Nie03, S.30ff] sind einige genannt: Ciphertext-Only, Known Plaintext, Chosen Plaintext, Chosen Ciphertext, Distinguishing Attacks, Birthday, Meet in the Middle.[2]

Ob und worauf Zero Knowledge anfällig ist, wird im folgenden Kapitel geklärt.

2.2 Historische Entwicklung

Die historische Entwicklung von Zero Knowledge (vgl. [Alb06, S. 378]) geht zurück auf das 16. Jahrhundert. Bereits in der Antike war das Interesse groß, Gleichungen 3. Grades[3] zu lösen. Nicolo Tartaglia (ca. 1500-1557) gelang es, eine Lösungsformel für diese Gleichungsart herauszufinden. Da Tartaglia jedoch in ärmlichen Verhältnissen lebte und keinen akademischen Titel besaß, beschloss er, die Formel geheim zu halten.

Durch einen Wettstreit mit Antonio Maria Fior, dessen Lehrmeister Scipione del Ferro bereits Jahrzehnte früher Lösungen für solche Gleichungen gefunden, jedoch nicht preisgegeben hatte, sollte nun Tartaglia seine Behauptung, diese Gleichungen lösen zu können, beweisen.

[1] Biometrische Verfahren werden zwar auch zur Identifizierung und Authentifizierung eingesetzt, jedoch ist der Einsatzbereich z.B. die Eintrittsberechtigung (wozu ZK natürlich auch verwendet werden kann). Zero Knowledge soll jedoch als unterliegende Sicherheitsmaßnahme gesehen werden. Weitere Überlegungen dazu sind in dieser Diplomarbeit zu finden.

[2] Diese genannten Attacken werden beispielsweise oft gegen Verschlüsselungsfunktionen eingesetzt.

[3] Gleichungen in der Form: $x^3 + ax^2 + bx + c = 0$

Hierzu gab Fior Tartaglia 30 Gleichungen, die dieser lösen sollte und auch lösen konnte. Somit war bewiesen, dass Tartaglia das Geheimnis kannte, ohne es preiszugeben. Und keiner seiner Konkurrenten konnte aufgrund der Lösung die Formel erraten.
Diese Art der Beweisführung ist das historisch erste Auftreten eines *interaktiven Beweises*. Tartaglia teile diese Formel Geronimo Cardana (1501-1576) mit, der diese, obwohl er geschworen hatte, diese nicht zu veröffentlichen, veröffentlichte. Jedoch erwähnte er in dieser "Ars Magna"Tartaglia explizit.

Tartaglias Methodik erfüllt nun die vorher genannten Eigenschaften, die Durchführbarkeit (er kann eine Lösung berechnen und diese kann verifizert werden) und Korrektheit (ohne die Formel kann die Lösung nur geraten werden, sehr kleine Wahrscheinlichkeit die richtige Lösung zu erraten). Desweiteren gilt auch die Zero Knowledge-Eigenschaft als erfüllt, da Tartaglia sein Wissen beweisen konnte, ohne das Geheimnis preiszugeben.
1985 wurde dieses Prinzip von Goldwasser, Micali und Rackoff in ihrem Werk *The Knowledge Complextity of Interactive Proof Systems*([Gol85]) erweitert und realisiert. Die Veröffentlichung dieses Werks war ein Meilenstein für die Weiterentwickung bzw. den Einsatz von Zero Knowledge.

2.3 Funktionsweise

Bei Zero Knowledge sind grundsätzlich zwei Akteure involviert, der Prover (Abk. P) und der Verifier (Abk. V).[4] Der Prover ist eine (manchmal versteckte) Entität, die den Beweis stellt und bestimmte Aussagen beweisen will. Der Verifier (bekannt) verifiziert die vom Prover gegebenen Antworten und legt großen Wert auf den Verifizierungs-Prozess.
Goldreich, Micali und Widgerson bezeichnen Zero Knowledge Beweise in [Ode91] als

Proofs that Yield Nothing But Their Validity [...]

Die Frage, die den Ursprung von Zero Knowledge bildet, ist, ob es möglich ist, eine Aussage zu beweisen, ohne Informationen darüber preiszugeben, außer ob diese Aussage richtig ist, oder nicht. Solche Beweise werden als zero-knowledge bezeichnet und spielen eine zentrale Rolle bei der Konstruktion von kryptographischen Protokollen.

[4]Prover und Verifier können in der Literatur auch als Peggy und Victor bezeichnet werden.

2.3.1 Mathematische/Interaktive Beweise

Definition Beweis (vgl. [Ode01, S. 187]): Ein Beweis in der Mathematik ist eine fixe Folge von Aussagen, welche entweder selbstverständlich oder von vorherigen (selbstverständlichen) Aussagen hergeleitet wurde. Mittlerweile ist man vom Begriff "selbstverständlich" hin zu "allgemein zugestimmt" übergegangen.
Zwei Eigenschaften von mathematischen Beweisen sind:

- Beweise werden als fixe Objekte gesehen
- Beweise werden mindestens so fundamental wie ihre Konsequenzen betrachtet

In anderen Bereichen des menschlichen Lebens kann man einen Beweis viel weiter ausgelegt sehen, wobei dieser kein fixes Objekt, sondern viel mehr ein Prozess ist, bei dem die Gültigkeit einer Aussage bewiesen wird (z.B. Gesetze, philosophische etc).
In der klassischen Mathematik haben Beweise eine statische Natur, in der Realität jedoch eine dynamische, d.h. sie erfolgen durch eine Interaktion.

2.3.2 Komplexitätsklassen

Komplexitätsklassen sind für Zero Knowledge grundlegend. Unter einer Komplexitätsklasse versteht man eine Menge von Problemen, die anhand ihrer Komplexität zusammengefasst werden können.
Bei ZK sind folgende Klassen von Bedeutung:

- P

  Die Komplexitätsklasse P beschreibt Probleme, die mit polynomialem Zeitaufwand lösbar sind. Dazu gehören Addition, Multiplikation, sowie Potenzieren und Sortieren.(vgl. [Alb06, S. 126f])

- NP

  Diese Problemklasse basiert auf der Verifizierung einer gegebenen Lösung. Hierbei haben die Probleme eine unterschiedliche Charakteristik (z.B. Faktorisierung großer Zahlen, Isomorphie von Graphen etc.).
  Laut [Alb06, S. 126] wird die Klasse wie folgt definiert:
  "Die Klasse NP besteht aus denjenigen Problemen, bei denen die Verifizierung einer

gegebenen Lösung mit polynomialem Zeitaufwand möglich ist."

Die Asymmetrie zwischen der Komplexität des Verifizierungsvorgangs und der Komplexität des Satz-bereitstellenden Vorgangs ist zusammengefasst durch diese Komplexitätsklasse, welche als Klasse der Beweissysteme gesehen werden kann. Jede Sprache $L \in NP$ hat einen effizienten Verifizierungsprozess für die Beweise von Aussagen der Form $x \in L$. (vgl. [Ode01, S.188], [Ode91, S. 693])

- IP

  Die Klasse IP wurde von Shamir bestimmt. In diese Klasse fallen z.B. alle mathematisch beweisbaren Sätze und auch das Finden einer Spielstrategie.

  Hierbei spielt der Ausdruck "IP=PSPACE" eine Rolle. IP ist hierbei die Klasse der interaktiv beweisbaren Sätze und PSPACE (polynomial space) die Klasse, welche Sätze, die mit polynomialem Speicherplatz-Aufwand aber evtl. exponentiellem Rechenaufwand bewiesen werden können. (vgl. [Alb06, S. 40])

Alle Probleme in NP lassen einen Zero Knowledge-Beweis zu

Eine grundsätzliche Eigenschaft bei dieser Thematik betrifft die Klasse NP. Hierbei beweisen Goldreich et al. ([Ode91, S. 698]), dass es für jedes Problem in NP einen Zero Knowledge-Beweis gibt.

Hierbei wird kein bestimmtes Schema verwendet, um sich einem geheimen Bit zu verschreiben (s. 2.3.4). Es werden Zero Knowledge-Beweissysteme für alle Sprachen in NP präsentiert. Ein Schema kann durch eine One-Way-Funktion implementiert werden. Es wird angenommen, dass solche One-Way-Funktionen existieren.

Durch diese Darstellung von Goldreich et al. wird die Allgemeingültigkeit von Zero Knowledge-Beweisen erklärt und somit der Schlüssel für den breiten Einsatz von Zero Knowledge in der Kryptographie.

2.3.3 Beispiel zur Erklärung von Zero Knowledge

Einfaches Beispiel

Die Funktionsweise von Zero Knowledge haben Guillou und Quisquater (vgl. [Sch96, S. 102f]) anhand einer Höhle dargestellt. Diese Höhle ist in Abbildung 2.1 dargestellt.

Es existiert in der Höhle eine Abzweigung, wobei die Enden durch eine geheime Tür verschlossen sind. Durch diese Tür können nur diejenigen hindurch gehen, welche das Geheimnis kennen.

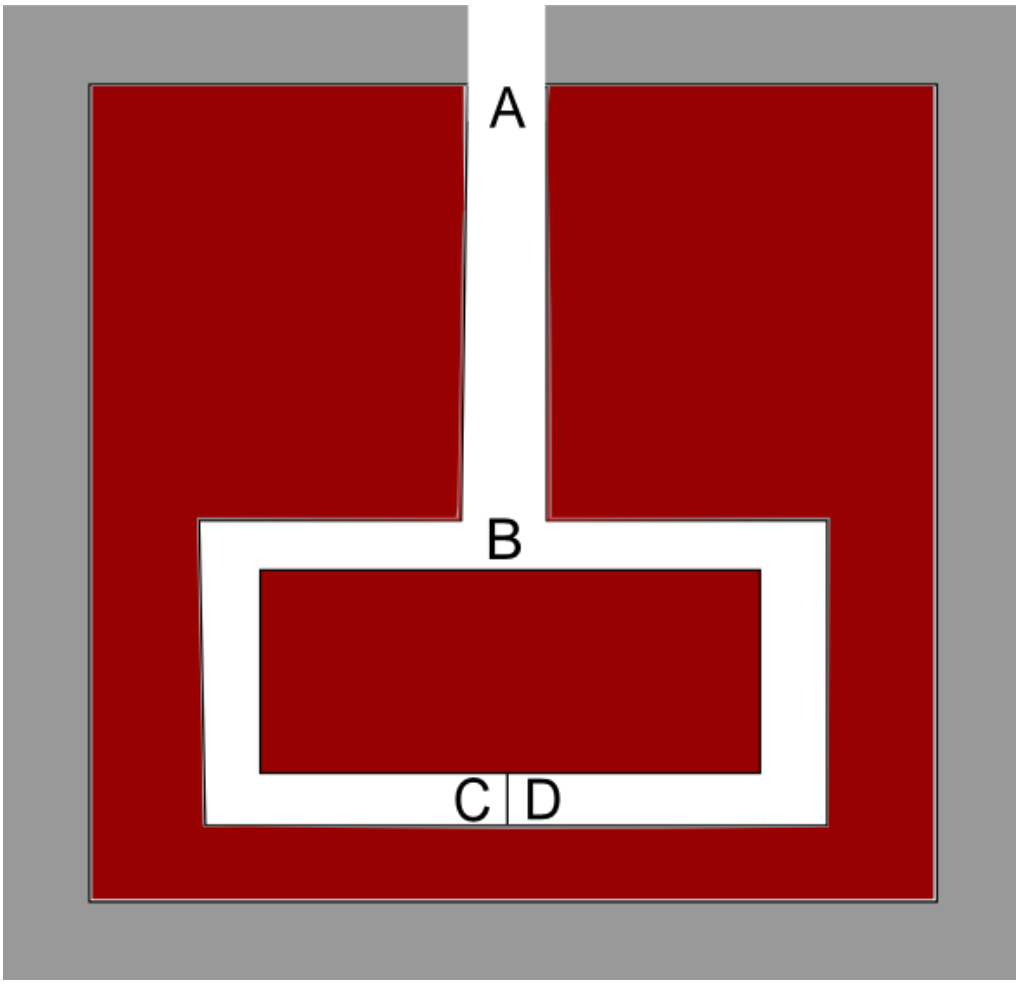

Abbildung 2.1: ZK Höhle

Peggy kennt das Passwort und möchte nun Victor beweisen, dass sie das Geheimnis kennt, ohne das Geheimis preiszugeben.
Dabei wird folgendermaßen vorgegangen:

1. Victor steht am Punkt A
2. Peggy geht durch die Höhle und befindet sich an Punkt C oder D
3. nachdem Peggy in die Höhle gegangen ist (Victor hat nicht mitbekommen, wohin genau), geht Victor zu Punkt B
4. nun möchte Victor, dass Peggy entweder links oder rechts herauskommt
5. Peggy erfüllt nun die Forderungen von Victor und öffnet, falls benötigt, die Tür mittels dem Geheimnis
6. Diese Schritte werden n Mal wiederholt

Auch wenn nun Victor diese n Vorgänge in irgendeiner Weise aufnehmen würde, könnte er Carol nicht davon überzeugen, dass Peggy das Geheimnis kennt, da diese Aufnahme in irgendeiner Weise gefälscht werden könnte oder im Vorhinein evtl. Absprachen getroffen worden sein könnten.
Resultierend daraus ist es unmöglich, einen Dritten davon zu überzeugen, dass der Beweis gültig ist. Und weiters ist dies Zero Knowledge, da Victor in keiner Weise (durch Betrachten der Aufnahmen) das Geheimnis herausfinden kann.
Bei dieser Protokollvariante wird das *Cut-and-Choose*-Prinzip angewendet (A schneidet die Sache auseinander, B wählt eine Hälfte für sich selbst und A nimmt den Rest).
Dieses Prinzip funktioniert in der Weise, dass Peggy Victor nur davon überzeugen kann, dass sie das Geheimnis kennt, wenn sie jedes Mal auf der richtigen Seite erscheint. Falls Peggy das Geheimnis nicht kennt, kann sie den Forderungen nicht folgen. Daher hat sie pro Runde eine 50% ige Chance, die Forderung von Victor durch Erraten zu erfüllen.

2.3.4 Grundlegendes Protokoll

Grundlegendes Protokoll
Das grundlegende Protokoll von Zero Knowledge kann in folgenden Schritten dargestellt werden (vgl. [Sch96, S. 103f]):

1. Peggy benutzt ihre (bekannte) Information und eine Zufallszahl, um das Problem in ein anderes Problem, welches isomorph zum Original ist, umzuwandeln. Mithilfe dieser Informationen kann sie dieses neue Problem lösen.

2. Peggy verpflichtet sich durch ein Bit-Commitment Scheme der Lösung

3. Peggy schickt Victor das neue Problem, wobei dieser nicht auf das Original schließen kann.

4. Victor wünscht nun entweder

   - den Isomorphismus zwischen dem originalen und dem neuen Problem
     oder
   - die Lösung für das neue Problem (welchem Peggy sich verpflichtet hat).

5. Peggy erfüllt den Wunsch von Victor.

6. Diese Schritte können n Mal wiederholt werden.

Exkurs: Bit-Commitment schemes

Commitment schemes (bzw. bit-commitment schemes) sind eine grundlegende Funktionalität von kryptografischen Protokollen. Diese können benutzt werden, um einen User an einen Wert zu binden und diesen geheim zu halten. Hierbei wird aber auch gesichert, dass der User den Wert wieder herausfinden kann. Zum besseren Verständnis kann man sich vorstellen, dass der Wert in einer Schachtel verschlossen ist. Diese Schachtel wird dem Empfänger übergeben, für den jedoch der Wert verschlossen ist, da er das Schloss nicht öffnen kann. Solange der Empfänger also die Schachtel hat, kann der Wert nicht verändert werden. (vgl. [Ode01, S. 223f])

2.3.5 Beispiel für ZK (Graphentheorie)

Ein weiteres Beispiel kommt aus dem Bereich der Graphentheorie.

2.3.5.1 Exkurs: Isomorphie von Graphen

Definition Graph ([Alb06, S. 122]): "Ein Graph besteht aus einer Menge von Ecken (Punkten), von denen je 2 durch eine Kante verbunden sind oder nicht. Jede Kante verbindet genau zwei Ecken."

Definition Isomorphie von Graphen ([Alb06, S. 122]): "Zwei Graphen werden isomorph (strukturgleich) genannt, wenn der eine Graph aus dem Anderen durch Umordnen der Ecken hervorgeht."
Zwei isomorphe Graphen werden in Abbildung 2.2 gezeigt.
Zwei Graphen heißen isomorph, falls es einen Isomorphismus zwischen ihnen, d.h. eine bijektive (eineindeutige) Abbildung der Menge der Ecken von G1 auf die Menge der Ecken von G2, sodass zwei Ecken von G2 genau dann durch eine Kante verbunden sind, wenn die Urbilder in G1 durch eine Kante verbunden sind (vgl. [Alb06, S. 122]). Dies impliziert natürlich, dass die Anzahl der Knoten als auch die Anzahl der Kanten gleich sein muss.
Die Erzeugung eines zu einem gegebenen Graphen G isomorphen Graphen, ist einfach, da man nur die Eckenmenge von G permutieren muss. Im Gegensatz dazu ist es jedoch sehr schwer, von zwei gegebenen Graphen zu entscheiden, ob diese isomorph sind bzw. einen Iso-

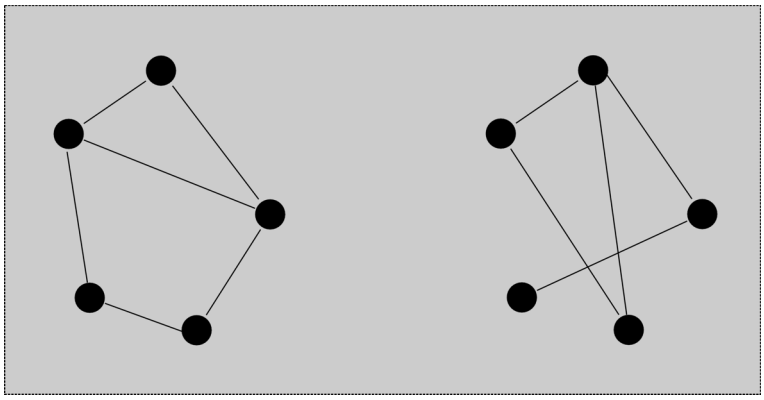

Abbildung 2.2: Zwei isomorphe Graphen

morphismus anzugeben. Wenn jedoch eine Permutation gegeben ist, kann einfach verifiziert werden, ob es ein Isomorphismus ist oder nicht.
Für einen ungerichteten Graphen G mit n Kanten gibt es n! isomorphe Graphen, d.h. für 2 hinreichend große Graphen G und G* ist es nicht möglich, in vernünftiger Zeit festzustellen, ob G und G* isomorph sind.

Anwendungsbeispiel Authentifizierung (vgl. [Alb06, S. 43])
Der Prover erzeugt Graph G0 und Isomorphismus σ, daraus einen Graph G1, wobei G1 zu G0 isomorph ist.
Die Graphen G0 und G1 werden öffentlich gemacht. Der Prover wählt G0 oder G1 aus, und wendet eine zufällige Permutation τ auf den gewählten Graphen an. Das Ergebnis ist ein Graph H, welcher an den Verifier geschickt wird.
Der Verifier wünscht entweder einen Isomorphismus zwischen H und G0 oder zwischen H und G1.
Der Prover schickt nun τ oder $\tau\sigma$ bzw $\tau\sigma^{-}1$ je nach Wahl des Graphen.
Der Verifier überprüft nun, ob die Abbildung ein Isomorphismus ist oder nicht.
Um nun diese Eigenschaft von Graphen auf die Zero Knowledge-Eigenschaften anzuwenden:

- Durchführbarkeit

  Der Prover kann sowohl $\tau\sigma$ als auch $\tau\sigma^{-}1$ bilden.

- Korrektheit

  Bei n Versuchen ist die Wahrscheinlichkeit $(\frac{1}{2})^n$ für einen erfolgreichen Betrug.

- ZK-Eigenschaft

  Hierbei ist die Simulation mit einem unwissenden Simulator möglich.

Der Verifizierungsvorgang ist relativ einfach (z.B. in polynomialer Zeit), wohingegen sich der Vorgang des Beweisens als schwierig herausstellen kann. Derjenige, welcher den Beweis tätigen will, hat die schwerere Aufgabe, da dieser auch z.B. mehr Rechenaufwand betreiben muss.
Zu erwähnen ist weiters, dass, falls der Prover in einem Vertrauensstand zum Verifier steht, kein Beweis notwendig ist.[5]

Die zwei fundamentalen Eigenschaften eines Beweissystems sind Soundness (oder Validität) und Completeness. Soundness bedeutet, dass beim Verifizierungsvorgang nicht betrogen werden kann, sodass falsche Aussagen akzeptiert werden. Diese Eigenschaft schützt den Verifier somit, dass dieser keine falschen Aussagen annimmt, wobei egal ist, was der Prover tut.
Completeness besagt, dass der Prover den Verifier überzeugen kann.
Beide Eigenschaften sind essentiell für Beweissysteme. Und die Effizienz eines solchen Beweissystems wird mit der Effektivität dessen Verfizierungsprozedur assoziiert.
Weiters gilt, dass die ZK-Eigenschaft nicht garantieren kann, dass ein Protokoll sicher ist. Genauso wenig kann dies die Soundness-Eigenschaft.

2.3.6 Interaktive Beweissysteme

Interaktive Beweissysteme sind die Grundlage für Zero Knowledge.
Tätigkeiten, welche bei Beweissystemen verwendet werden, sind:

- Erstellung des Beweises
- Verifizieren der Validität des Beweises

Diese Tasks werden durch Prover und Verifier durchgeführt, welche miteinander interagieren.
Bei manchen Fällen kann die Interaktion relativ einfach (z.B. unidirektional - Prover schickt Text an Verifier) abfolgen, grundsätzlich jedoch ist die Interaktion komplexer (z.B. Verifier befragt den Prover).
Hierbei ist Voraussetzung, dass der Verifier korrekte Aussagen mit hoher Wahrscheinlichkeit

[5] Trotzdem werden, falls über Beweise diskutiert wird, diese als nicht vertrautswürdig gelten, d.h. der Verifier immer skeptisch gegenüber jeder Aussage des Provers sein.

annimmt bzw. die Wahrscheinlichkeit, dass dieser falsche Aussagen annimmt, sehr gering ist (egal, womit interagiert wird).
Prover und Verifier tauschen mehrfach Nachrichten aus (Challenge und Response), welche auf Zufallszahlen basieren (idealerweise fairer Münzwurf). Der Prover muss den Verifier überzeugen, dass eine Aussage der Wahrheit entspricht. Der Verifier entweder akzeptiert oder verweigert den Beweis. Die traditionelle mathematische Notation eines Beweises ist in dem Sinne verändert worden, dass diese nicht absolut, sondern eher probabilistisch sind. Hierbei muss der Beweis nur innerhalb eines gewissen Wahrscheinlichkeitsbereichs liegen (z.B. nahe 1). Ein interaktiver Beweis wird manchmal als *Proof by Protocol* bezeichnet. Interaktive Beweise, die für die Identifikation verwendet werden, könnten als *Proofs of Knowledge* bezeichnet werden. (vgl. [Alf96, S. 406])

2.3.6.1 Interaktive Beweissysteme und die Komplexitätsklasse IP

Ein interaktives Beweissystem für ein Set S ist ein 2-Parteien-Austausch zwischen einem Verifier, der eine probabilistische Zeit-polynomische Strategie V ausführt, und einem Prover, der eine nicht an eine Berechnung gebundene Strategie P ausführt, wobei die Eigenschaften

- Completeness
  Für jedes $x \in S$ akzeptiert der Verifier V immer die Interaktion mit dem Prover P am gemeinsamen Input x.
- Soundness
  Für ein polynomes p, jedes $x \notin S$ und jede potentielle Strategie P* verweigert der Verifier mit der Wahrscheinlichkeit von mind. $\frac{1}{p[|x|]}$ nach der Interaktion mit P* am gemeinsamen Input x.

erfüllt werden. (vgl. [Ode04, S. 5])

Die Komplexitätsklasse von Problemen, welche ein interaktives Beweissystem besitzen, wird als IP bezeichnet.
Weiters wird hier erwähnt, dass unter der selben Aussage, die bei NP genutzt wird, jedes Set, das einen interaktiven Beweis besitzt auch einen Zero Knowledge-interaktiven Beweis hat.
Bei der Definition von ZK-Beweisen wird der Verifier als ein potentieller Angreifer gesehen,

der Wissen vom Prover erhalten will.
Ein interaktiver Beweis, welcher ein Challenge-Response-Protokoll darstellt, besteht aus einer spezifizierten Anzahl von Runden (vgl. [Sti95]):

1. Erhalten einer Nachricht vom anderen Teilnehmer
2. Durchführen einer eigenen, privaten Berechnung
3. Senden einer Nachricht an den anderen Teilnehmer

Typischerweise sieht eine Runde wie folgt aus: Challenge vom Verifier, Response vom Prover. Am Ende verweigert der Verifier oder aktzeptiert, je nachdem ob der Prover richtig antwortet.
In einer kryptographischen Umgebung hat ein User U ein Geheimnis und wird normalerweise darauf basierend eine Handlung tätigen. Wie kann nun ein anderer User verifizieren, dass U das Richtige getan hat (durch U's Geheimnis und öffentliche Information). Wenn U das Geheimnis offenlegen würde, könnte jeder dieses verifizieren. Mit Zero Knowledge-Beweisen kann dieses verifiziert werden, ohne das Geheimnis offenzulegen. U's Behauptung, die richtige Aktion getan zu haben, ist eine NP-Aussage (polynomische Zeit-Funktion des Geheimnisses und der öffentlichen Informationen) und dass U eine NP-Herausforderung betreffend seiner Gültigkeit hat (z.B. das Geheimnis ist eine NP-Aussage, um zu bestätigen, dass die Aktion zur öffentlichen Information passt). Daher ist es für U möglich, die Richtigkeit seiner Aktion darzustellen, ohne das Geheimnis preiszugeben. (vgl. [Ode05, S. 42f])

2.3.7 Unterteilung von Zero Knowledge

Die Unterteilung von ZK basiert auf einem Simulator. Dieser kann je nach Ähnlichkeit der Simulationen zum realen Ablauf (d.h. z.B. der Output vom Verifier nach der Interaktion mit dem Prover und der Output des Simulators) folgendermaßen eingeteilt werden (vgl. [Sch96, S. 108], [Ode05, S. 45f]):

1. Perfect Zero Knowledge

   Perfect Zero Knowledge setzt voraus, dass die Ergebnisse identisch sind.

2. Statistical Zero Knowledge

   Statistical Zero Knowledge setzt voraus, dass diese Wahrscheinlichkeiten statistisch na-

he sind (Variationsdistanz ist unerheblich) bzw. dass diese bis auf wenige Ausnahmen ident sind.

3. Computational Zero Knowledge
   Computational Zero Knowledge (bzw. "normales" ZK) setzt voraus, dass diese Wahrscheinlichkeiten von der Berechnung her nicht unterscheidbar sind.

4. No-Use
   Ein Simulator kann existieren, jedoch kann bewiesen werden, dass der Verifier keine Information des Beweises mitbekommt.

Diese Einteilung ist zwar nicht konkret relevant für diese Arbeit, jedoch eine Erwähnung wert.

2.3.8 Grundlegende Probleme

Bei Zero Knowledge können unter anderem auch folgende Probleme auftreten (vgl. [Ode05, S. 53ff]):

- Composition
  Composition besagt, dass die Security unter verschiedenen Composition-Operationen bestehen bleibt. Hierbei sind unabhängige Ausführungen eines Prokolls gemeint, die durch einen Angreifer attackiert werden.

- Black-Box Simulation
  Eine Black-Box ist ein universeller Simulator, wie z.B. ein Orakel. Negative Resultate bedeuten hierbei eine Limitation von Zero Knowledge. Barak hat in [Bar01] mithilfe der richtigen Eigenschaftenwahl diese Barriere überwunden, sodass einer Black-Box Simulation standgehalten wird.

2.3.8.1 Composition

- Sequentielle Composition
  Zero Knowledge ist bei sequentieller Composition in sich geschlossen.

- Parallele Composition
  Zero Knowledge ist bei paralleler Composition nicht in sich geschlossen, wobei jedoch

manche Zero Knowledge-Beweise (für NP) die Sicherheit behalten, wenn mehrere Vorgänge parallel abgehandelt werden. Manche dieser Protokolle verwenden eine konstante Anzahl an Runden bei der Durchführung.

- Concurrent Composition
  Concurrent Composition bedeutet, dass ZK völlig asynchron ausgeführt wird. D.h. dass diese im Vergleich zu paralleler Composition keine konstante Rundenanzahl verwenden und diese Protokolle ihre Sicherheit voll bewahren, wenn diese parallel ausgeführt werden.
  In [Ros06] wird gerade diese Thematik speziell behandelt.

- Rundenkonstante ZK-Beweise im limited asynchronous Modus
  Hierbei verwendet jeder Akteur eine lokale Uhr, die an eine vorher bekannte Konstante gebunden ist. Somit können zeitgesteuerte Operationen durchgeführt werden.

2.3.9 Parallele Ausführung

Eine grundsätzliche Frage betreffend Zero Knowledge-Protokollen bezieht sich auf die parallele Ausführung des Protokolls von mehreren Instanzen (simultan bzw. asynchron).
Dass die praktische Bedeutung dieser Frage heutzutage bei beträchtlicher Internet- bzw. Kommunikationsnutzung eine große Rolle spielt, sollte klar sein. Diese Frage stellt auch eine interessante Basis für die theoretische Betrachtung der Herausforderung, die Security bei paralleler bzw. gleichzeitiger Ausführung von kryptografischen Protokollen zu bewahren und kann als Benchmark eingesetzt werden. In diese Richtung gehen mittlerweile mehrere Entwicklungen in der Kryptographie, wie man mit diesem Problem umgehen kann.
Bei parallelen ZK-Beweisen (vgl. [Sch96, S. 106]) findet zwischen Prover und Verifier n Mal ein Informationsaustausch statt.

1. Der Prover nimmt die Informationen und n Zufallszahlen, um das Problem in n isomorphe Probleme zu transformieren. Diese werden nun verwendet, um diese n neuen Probleme zu lösen.

2. Der Prover bindet sich an diese neuen Probleme.

3. Diese Probleme werden nun an den Verifier geschickt.

4. Für jedes dieser Probleme kann der Verifier nun wählen, ob er von Prover

- die Isomorphie zwischen den alten und neuen Problemen
  oder
- die Lösung vom Problem bzw. den Beweis für die Lösung

möchte.

5. Für jedes der n Probleme wird eine Antwort geliefert.

Im Vergleich zu sequentiellen ZK-Protokollabläufen kann bei paralleler Ausführung nicht garantiert werden, dass die ZK-Eigenschaften erhalten bleiben. Die ZK-Eigenschaft bleibt zwar erhalten, jedoch kann dies in gewisser Weise nicht bewiesen werden.
Weitere wichtige bzw. interessante Begriffe in Bezug auf die parallele Ausführung von ZK-Protokollen sind *Witness Hiding* und *Witness Indistinguishability*. (vgl. [Alb06, S. 54ff])
Bei der parallelen Ausführung von ZK-Protokollen muss die ZK-Eigenschaft nicht immer erhalten bleiben und das Protokoll gilt noch immer als sicher. Hierbei werden von P und V nicht nur eine, sondern mehrere Nachrichten gesendet. Dadurch wird die Anzahl der zu sendenden Nachrichten klein, was z.B. für dein Einsatz bei Chipkarten von Vorteil ist, da der Übertragungsweg der Flaschenhals ist.
Durch das Witness-Hiding-Protokoll und den Begriff Witness Indistinguishability kann gezeigt werden, dass die WH-Eigenschaft bei beliebig komplexen Protokollen erhalten bleibt. Hierbei kann ein Verifier jedoch möglicherweise Informationen mitbekommen. Über das spezielle Geheimnis vom Prover erfährt er jedoch nichts.
Bei dieser Eigenschaft wird ein zweiter Zeuge eingeschalten, und der Verifier kann nicht unterscheiden, welcher nun verwendet wurde bzw. welches Geheimnis dieser besitzt. Hierdurch können, wie bereits erwähnt, Informationen betreffend der Menge aller Witnesses eventuell mitbekommen werden, jedoch nicht unterschieden werden, welchem Zeugen welche Information zugeordnet ist. Somit bleibt die Übertragung sicher.

2.3.10 Nicht-interaktive Zero Knowledge-Beweise

Nicht-interaktive ZK-Beweise Für nicht-interaktive ZK-Beweise (vgl. [Sch96, S. 106ff]) wurden bereits Protokolle entwickelt, um jemand anderen (nicht beteiligten), den der Beweis interessiert, davon zu überzeugen, dass der Beweis gültig ist.
Hierbei ist dies ähnlich zur parallelen Ausführung, jedoch wird der Platz vom Verifier durch eine Einweg-Hashfunktion ersetzt.

1. Der Prover nimmt die Informationen und n Zufallszahlen, um das Problem in n isomorphe Probleme zu transformieren. Diese werden nun verwendet, um diese n neuen Probleme zu lösen.

2. Der Prover bindet sich an diese neuen Probleme.

3. Der Prover nimmt alle diese Commitments von Schritt 2 als Input für eine One-Way-Hashfunktion. Die ersten n Bits des Outputs der Hash-Funktion werden gespeichert.

4. Die n gespeicherten Bits werden für jedes i-te Problem verwendet, und jedes i-te Bit dieser n Bits wird je nachdem verwendet:

   - b = 0: es wird die Isomorphie gezeigt
   - b = 1: Die Lösung von Schritt 2 wird geöffnet und bewiesen, dass dieses eine Lösung zum neuen Problem ist.

5. Der Prover publiziert alle Commitments vom Schritt 2 sowie die Lösungen von Schritt 4.

6. Die weiteren Interessierten können nun dies verifizieren.

Dadurch kann der Prover Daten publizieren, ohne etwas vom Geheimnis zu verraten und trotzdem jemanden davon zu überzeugen, dass das Geheimnis existiert. Dieses ”Protokoll” kann auch z.B. für digitale Signaturen verwendet werden (s. [Sch96, S. 207]).
In diesen nicht-interaktiven Protokollen müssen mehrere Iterationen durchgeführt werden als bei interaktiven Protokollen. Vergleichend zu obigem Ablauf kann ein Angreifer bei Schritt 4 jeweils eine der Varianten auswählen und verifizieren, ob dies korrekt war. Falls nicht, kann der Ablauf einfach wiederholt werden.

2.3.11 Effizienz-Messungen

Folgende Parameter können für die Effizienz-Messung eingesetzt werden(vgl. [Ode01, S. 244f]):

- Communication complexity
  Die Communication complexity beschreibt die Kommunikation beim Beweisvorgang.

Die hierbei wichtigste Eigenschaft ist die Round complexity, sprich die Anzahl von ausgetauschten Nachrichten.

- Computational complexity
  Die Computational complexity ist die Nummer von grundlegenden Schritten, die von jedem Beteiligten durchgeführt werden.

- Round complexity
  Die Round Complexity eines Protokolls ist eine wichtige Effizienz-Variable und es ist wichtig, diese zu minimieren. Bei Zero Knowledge-Protokollen ist es wichtig, diese konstant zu halten und die Fehlerwahrscheinlichkeit so klein als möglich zu halten.

- Knowledge tightness
  Die vorherigen Eigenschaften sind generisch und somit auf jedes Protokoll anwendbar. Knowledge tightness ist jedoch spezifisch auf Zero Knowledge anwendbar. Der Sinn dieser Eigenschaft ist, die ”effektive” Sicherheit eines Beweissystems zu verifizieren, also wie viel härter der Verifier ohne Interaktion mit dem Prover ”arbeiten” (also z.B. rechnen) muss. Die Berechnung erfolgt als Rate zwischen der (erwarteten) Laufzeit des Simulators und der Laufzeit des Verifiers in einer ”realen” Interaktion (welche durch den Simulator simuliert wird). Diese Rate sollte in etwa gleich bleiben.

Ein Verifier, der den Beweis erhält, bekommt nur die Gültigkeit der Aussage (als wäre dieser von einer vertrauten Stelle gekommen).

Es ist bekannt, dass ZK-Beweise für jedes Set in NP existieren, wenn one-way Funktionen existieren. Daher kann diese Eigenschaft wesentlich für das Design von kryptografischen Protokollen eingesetzt werden, weil man die Akteure zwingen kann, sich an ein definiertes Protokoll zu halten (z.B. ZK-Beweise).

Moderne Kryptographie beschäftigt sich mit der Konstruktion von effizienten Schemata, welche es unmöglich bzw. praktisch nicht durchführbar machen, die Security zu schwächen. Dasselbe Problem ist die Grundlage von Zero Knowledge.
Zufällige Berechungen spielen bei ZK eine zentrale Rolle (wie grundsätzlich überall bei der Kryptographie). Hierbei wird einem erlaubten User gestattet, eine zufällige Berechnung durchzuführen, sowie auch einem Angreifer. Das Ziel dafür ist, dass der erlaubte User mit einer annähernd 100% igen Wahrscheinlichkeit erfolgreich ist und die Angreifer nur mit einer

sehr kleinen Wahrscheinlichkeit Erfolg haben.

2.3.12 Zusammenfassung

Die Forschung im Bereich Zero Knowledge wurde seit den 1980er-Jahren intensiviert, um theoretische und praktikable Lösungen zu finden. Weiters wurde auch erforscht, dass die ZK-Eigenschaft beweisbar ist.
Zero Knowledge kann als Identitätsnachweis bzw. als Authentifizierungsmethodik gesehen werden. Die Anwendungsbereiche sind z.B. bei Chipkarten, Pay-TV etc. (mehr dazu in Kapitel 4). Mithilfe von Zero Knowledge kann sich ein User legitimieren, ohne einen Schlüssel o.Ä. zu übertragen, d.h. auch auf z.B. Passwörter kann verzichtet werden.
Laut [Ros06] ist die Einsatzfähigkeit von Zero Knowledge-Verfahren im Bereich der Kryptographie enorm. Diese werden oft dazu genutzt, um böswillige Angreifer dazu zu bewegen, sich an ein vordefiniertes Protokoll zu halten.
Nicht nur durch die direkte Anwendung in der Kryptographie, sondern auch als Benchmark für Probleme bei kryptographischen Protokollen sind Zero Knowledge-Beweissysteme ein gutes Mittel.
Die Ausführung von Zero Knowledge-Protokollen muss garantieren, dass keine Informationen preisgegeben werden, daher ist klarerweise auch das Thema der Parallelität beim Ablauf von Relevanz. Um diese gewährleisten zu können, kommen verschiedene Protokolle (wie folgt) oder Abwandlungen davon zum Einsatz.

2.4 Protokolle

2.4.1 Generelle Struktur

Die generelle Struktur eines Zero Knowledge-Protokolls sieht wie folgt aus:

$$P \Rightarrow V\text{: witness}$$
$$P \Leftarrow V\text{: challenge}$$
$$P \Rightarrow V\text{: response}$$

Der Prover selektiert ein zufälliges Element aus einem vordefinierten Set als sein Geheimnis (”versteckte Zufälligkeit” bzw. ”privater Münzwurf”). Dadurch wird beim Start eines

Protokolllaufs eine Zufälligkeit garantiert. Nach dem Design eines Protokolls kann nur ein legitimer Prover, der das Geheimnis wirklich kennt, alle Fragen beantworten und gibt dabei keine Informationen darüber preis.
Das Protokoll kann mehrfach iteriert ausgeführt werden, um die Wahrscheinlichkeit zu betrügen zu minimieren.
Zero Knowledge-Protokolle vereinen die Ideen vom *Cut-and-Choose*-Prinzip (einer schneidet, der andere wählt) und von Challenge-Response-Protokollen.

Von Relevanz sind die nun folgenden Protokolle (vgl. [Alf96, S. 408ff]).

2.4.2 Protokoll 1 - Fiat-Shamir

Das Fiat-Shamir-Protokoll ist die Basisvariante, welche die Idee von Zero Knowledge darstellen soll. In der Praxis sollten bzw. werden effizientere Protokolle, wie z.B. das Feige-Fiat-Shamir-Protokoll, verwendet.
Das Ziel ist, dass sich der Prover identifiziert, indem er das Wissen eines Geheimnisses s dem Verifier beweist, ohne dass jedwegliche Information über s bei der Abhandlung des Protokolls preisgegeben wird.
Die Security beim Fiat-Shamir-Protokoll basiert auf der Schwierigkeit, Quadratwurzeln modulo große Integer (unbekannter Faktorisierung) zu berechnen.

2.4.2.1 Durchführung

Die folgenden Schritte werden beim Fiat-Shamir-Protokoll durchgeführt (vgl. [Alf96, S.408f]):

1. P wählt eine zufällige Zahl r, wobei $1 \leq r \leq n-1$ und schickt $x = r^2 mod n$ an V
2. V wählt zufällig ein (Challenge-)Bit $e = 0$ oder $e = 1$ und schickt e an P
3. P berechnet und sendet V die Antwort y, wobei basierend auf e: $e = 0 : y = r$ oder $e = 1 : y = rsmodn$
4. V verweigert den Beweis falls $y = 0$ und andererseits akzeptiert, nachdem verifiziert wurde, dass $y^2 \equiv x * v^e (mod n)$(abhängig von $e, y^2 = x$oder $y^2 = xvmodn$, da ja $v = s^2 modn$)

Die Aufgabe e benötigt, dass P zwei Fragen beantworten kann. Eine, die das Wissen des Geheimnisses demonstriert, und die zweite (eine einfache Frage), die verhindern soll, dass betrogen werden kann.

2.4.3 Protokoll 2 - Feige-Fiat-Shamir

Das Feige-Fiat-Shamir Identifikations-Protokoll bezieht eine Entität mit ein, die sich selbst durch einen Zero Knowledge-Beweis identifiziert.
Dieses Protokoll benötigt weniger Berechnungsaufwand und ist deshalb gut für Applikationen mit low-power Prozessoren (z.B. Chipkarten-Mikroprozessoren) geeignet.

2.4.3.1 Durchführung

Die folgenden Schritte werden t-Mal wiederholt. Der Verifier akzeptiert die Identität vom Prover, falls alle t Wiederholungen erfolgreich sind.
Voraussetzung dafür ist, dass der Verifier den public key vom Prover besitzt. Falls dieser nicht vorhanden ist, wird bei der ersten Nachricht ein Zertifikat mitgeschickt.

1. P wählt einen zufälligen Integer $r, 1 \leq r \leq n-1$, und ein zufälliges bit b und berechnet $x = (-1)^b * r^2 mod n$ und sendet x (Witness) an V
2. V sendet P einen k-bit Vektor (e1,...ek) als Aufgabe
3. P berechnet und sendet B die Antwort: $y = r * \prod_{j=1}^{k} s_j^{e_j}$
4. V berechnet $z = y^2 * \prod_{j=1}^{k} v_j^{e_j}$ und verifiziert, dass $z = \pm x$ und $z \neq 0$

2.4.4 Protokoll 3 - Guillou-Quisquater

Das Guillou-Quisquater (GQ) Identifikations-Schema ist eine Erweiterung des Fiat-Shamir Protokolls. Das GQ-Schema verringert sowohl die Anzahl der versandten Nachrichten, als auch die Speicheranforderungen für die Geheimnisse und ist somit auch geeignet für Geräte mit limitierten Spezifikationen.
Bei GQ werden drei Nachrichten zwischen P und V gesandt, um die Identität von P zu bestätigen.

2.4.4.1 Durchführung

Folgende grundsätzliche Variablen werden verwendet (teilweise von einem Trustcenter festgelegt):

- $\phi : (p-1)(q-1)$
- v: öffentlicher Exponent
- s: privater Exponent, $s = v^{-}1 mod \phi$
- $J_A = f(I_A)$

1. P wählt einen zufälligen geheimen Integer $r, 1 \leq r \leq n-1$, und berechnet $X = r^v mod n$
2. P sendet V das Integerpaar (I_A, x), wobei I_A eine eindeutige Identität darstellt
3. V wählt einen zufälligen Integer und schickt diesen an P, wobei$1 \leq e \leq v$
4. P berechnet und schickt V die Response $y = r * s_A^e mod n$
5. V erhält y, erstellt J_A aus I_A mittels f, berechnet $z = J_A^e * y^v mod n$ und akzeptiert den Beweis von P falls beide $z = x$ und $z \neq 0$

2.4.5 Protokoll 4 - Schnorr

Das Schnorr Identifikations-Prokoll ist eine Alternative zu Fiat-Shamir und GQ. Die Sicherheit dieses Protokolls basiert auf dem Problem des diskreten Logarithmus. Das Design erlaubt Vorberechnungen, sodass weniger Berechnungen in Echtzeit durchgeführt werden müssen. Daher ist dieses Protokoll teilweise für Geräte mit weniger Rechenleistung verwendbar. Es wurde für drei Nachrichtenübertragungen und eine niedrige Übertragungsbandbreite konzipiert.

2.4.5.1 Durchführung

$\beta : 1 \leq \beta \leq p-1$
$a : privatekey, 0 \leq a \leq q-1$

1. P wählt ein zufälliges $r, 1 \leq r \leq q-1$, berechnet $x = \beta^r mod p$ und schickt $cert_A$ und x an V
2. V authentisiert den public key v von P durch Verifizierung der Signatur von T auf $cert_A$, dann wird ein zufälliges $e, 1 \leq e \leq 2^t$ an P gesandt.
3. P überprüft, ob $1 \leq e \leq 2^t$ und schickt die Antwort $y = ae + r mod q$ an V.
4. V berechnet $z = \beta^y v^e mod p$, und akzeptiert die Identität von P, falls $z = x$.

2.4.6 Bewertung der Protokolle

2.4.6.1 Fiat-Shamir

Das Fiat-Shamir-Protokoll ist eine sehr einfache Implementierung von ZK und erfüllt alle Eigenschaften. Die dafür erforderlichen Berechnungen sind einfach und können z.B. auf einer Chipkarte durchgeführt werden. Im Vergleich zu RSA ist die Anzahl der Schritte 1-2 Mal kleiner.
Durch die hierbei gesendeten Informationen kann das Geheimnis nicht herausgefunden werden, da die modulo-Berechnung einer Quadratwurzel nicht möglich ist bzw. da p und q nicht bekannt sind.
Die Wahrscheinlichkeit, dass betrogen werden kann bzw. dass richtig geraten werden kann beträgt pro Runde 50%. D.h. falls immer vorhersehbar wäre, welches Bit gewünscht ist, könnte betrogen werden. Da jedoch die Wahrscheinlichkeit, dass immer richtig geraten wird und die Authentifizierung nur angenommen wird, wenn alle Antworten richtig sind, ist die Betrugswahrscheinlichkeit minimal. (vgl. [Alf96, S. 409 bzw. S. 416f])

2.4.6.2 Feige-Fiat-Shamir

Das Feige-Fiat-Shamir-Protokoll (FFS-Protokoll) dient zur Identifizierung und entstand durch die Abänderung des Fiat-Shamir-Protokolls.
Das FFS-Protokoll benötigt ebenfalls wenige Berechnungen und kann somit auch z.B. auf Chipkarten eingesetzt werden.
Die Wahrscheinlichkeit, dass betrogen wird, kann durch den Fakt, dass die Faktorisierung großer Zahlen schwierig ist, als nichtig dargestellt werden.
Um dieses Protokoll im Einsatz effektiv zu gestalten, müssen die verwendeten Parameter gut

gewählt werden, sodass ein gutes Mittelmaß aus Sicherheit und Berechnungszeit hervorgeht. (vgl. [Alf96, S. 411])

2.4.6.3 Goillou-Quisquater

Das GQ-Protokoll (Goillou-Quisquater-Protokoll) ist vom FFS-Verfahren abgeleitet und erlaubt ebenfalls den Einsatz auf leistungsschwächeren Geräten, da die Anzahl der gesendeten Nachrichten und der Speicherbedarf minimiert werden können.
(vgl. [Alf96, S. 413])

2.4.6.4 Schnorr

Das Protokoll nach Schnorr basiert auf dem Problem der Berechnung von diskreten Logarithmen. Es ist jedoch hierbei möglich Vorberechnungen anzustellen, um die Berechnungen in Echtzeit zu verringern. Daher ist eine Möglichkeit der Einsatz auf schwachen Geräten, jedoch muss aufgrund dessen die Vorberechnung praktikabel sein. Hierbei müssen jedoch die Parameter gut gewählt werden, um ein brauchbares Ergebnis zu erzielen und z.B. die Anzahl der übertragenen Daten klein zu halten. (vgl. [Alf96, S. 416])

Der Einsatz dieser genannten Protokolle basiert immer auf dem gewünschten Einsatzbereich. Wie zuvor erwähnt, sind Eigenschaften wie Berechnungszeit, Berechnungsaufwand, Speicherung bzw. Bufferung relevant und können die Funktion bzw. den Einsatz beschränken oder ebenfalls erweitern. Auch Anpassungen für den Einsatz der Protokolle können getroffen werden und die Parameter spezifisch gewählt werden. Ebenfalls sind Abwandlungen der genannten Protokolle nicht ausgeschlossen bzw. sogar sinnvoll, wenn diese für spezielle Applikationen eingesetzt werden sollen.

2.5 Challenge-Response

Hinter kryptographischen Challenge-Response-Protokollen (vgl. [Alf96, 397ff]) steckt die Idee, dass eine Entität ihre Identität einer anderen durch das Wissen eines Geheimnisses beweist. Dieses Geheimnis ist mit dieser Entität assoziiert, und soll während des Protokollablaufs nicht offenbart werden.

Dies geschieht durch eine zeitvariante Aufgabe, wobei die Antwort darauf vom Geheimnis und dieser Aufgabe abhängt.
Die Aufgabe ist normalerweise eine Zahl, die zufällig und geheim außerhalb des Protokolls gewählt wird. Falls die Kommunikation überwacht wird, sollte keine verwendbare Information über die Identifikation preisgegeben werden.

2.5.1 Hintergrund

Zeit-variante Parameter können bei Identifikations-Protkollen benutzt werden, um z.B. Replay-Attacken oder Interleaving Attacks zu verhindern. Weiters können diese eingesetzt werden, um eine Eindeutigkeit zu gewährleisten oder verschiedene Chosen-Text-Attacken zu verhindern.(vgl. [Alf96, S. 397])
Um verschiedene Instanzen von Protokollen unterscheiden zu können, werden z.B. Nonces, d.h. eindeutige Nummern oder nicht-wiederholende Werte verwendet. Timestamps z.B. benötigen weniger Nachrichten und auch keine long-term Statusinformation (wie Sequence Numbers, etc).
Das Minimieren von Statusinformationen ist teilweise wichtig für die Server in Client-Server Applikationen.
Timestamps werden in Protokollen oft durch eine Aufgabe mit Zufallszahl und einer Return-Message ersetzt.(vgl. [Alf96, S. 400])
Das Challenge-Response Verfahren basiert auf Techniken mit symmetrischen Schlüsseln (die symmetrische Variante). Diese setzen voraus, dass ein symmetrischer Schlüssel geteilt wird. Für geschlossene Systeme mit wenigen Usern wird pro Userpaar ein Schlüssel geteilt. Bei größeren Systemen wird oft ein vertrauenswürdiger Server für die Verwaltung der Schlüssel eingebunden. Dieser Server stellt einen gemeinsamen Session Key zur Verfügung, wenn sich zwei Parteien authentifizieren wollen.
Challenge-Response basiert auf One-way Funktionen, wobei der Verschlüsselungsalgorithmus durch eine One-way Funktion oder eine nicht reversible Funktion eines Schlüssels und eine Challenge (wie z.B. MAC) sein kann. Diese Funktionalität kann verwendet werden, wenn z.B. eine Verschlüsselung wichtig ist (z.B. wegen des Berechnungaufwandes).
Die Beantwortung einer Challenge bei Challenge-Response Protokollen benötigt ein Berechungsdevice und eine sichere Speichereinheit für das Langzeit-Schlüsselmaterial (z.B. Datei auf vertrauter Platte etc.). Für mehr Sicherheit kann z.B. eine Chipkarte für die Schlüsselspeicherung und Berechnung verwendet werden. Diese Variante ist günstiger als z.B. ein

Passcode Generator (Devices, die Passwörter oder Passcodes mithilfe eines gerätespezifischen geheimen Schlüssels erstellen).(vgl. [Alf96, S. 402ff])

Challenge-Response durch public-key Techniken(vgl. [Alf96, S. 403])
Public-Key Techniken werden für die Identifikation verwendet, wobei eine Entität das Wissen eines geheimen Schlüssels beweist. Dies geschieht folgendermaßen:

- Entschlüsseln einer Challenge, welche mit dem public Key verschlüsselt wurde oder
- Eine Challenge wird digital signiert

2.5.2 Funktionsweise

Die grundsätzliche Funktionsweise von Challenge-Response wird hier nun dargestellt.(vgl. [Alb06, S. 24])
Hierbei wird folgender Struktur gefolgt:
User und PC besitzen bei z.B. symmetrischer Variante einen gemeinsamen Schlüssel. Dieser wird durch ein Identifikationsmerkmal (z.B. Username) abgefragt und ausgelesen. Nun muss sich der PC überzeugen, dass der User den Schlüssel besitzt, ohne dass der Schlüssel übertragen wird. Der Vorgang läuft folgendermaßen ab:

- Beide Beteiligten verfügen über einen gemeinsamen Algorithmus (welcher eine One-Way-Funktion sein sollte, sodass der Schlüssel nicht berechnet werden kann)
- PC schickt eine Zufallszahl an den User
- User berechnet Antwort mithilfe des Algorithmus und schickt diese zurück
- PC bildet diese Antwort selbst und vergleicht die Antworten
- Falls die Antworten gleich sind, ist der User höchstwahrscheinlich authentisch

2.5.3 Vergleich mit Zero Knowledge

Zero Knowledge ist ein Verfahren zur Identifikation, welches grundsätzlich asymmetrische Verfahren verwendet, aber nicht auf digitalen Signaturen, Sequence Numbers oder public-key Verschlüsselung basiert und keine Block Ciphers, Timestamps etc. verwendet.

Einfache Passwort-Protokolle haben einen großen Nachteil. Der Prover gibt dem Verifier das Passwort. Challenge-Response Protokolle verbessern dies durch eine Challenge, um zu beweisen, dass ein Geheimnis bekannt ist, wobei jedoch garantiert wird, dass keine direkt verwertbare Information für z.B. eine Angreifer vorhanden ist. Jedoch könnte ein angreifender Verifier strategisch Challenges auswählen um Informationen zu erhalten (Chosen-Text Attack, ...).
Zero Knowledge-Protokolle wurden dafür kreiert, um diese Probleme zu behandeln, und überhaupt keine Information preiszugeben.
Bei Zero Knowledge wird nur ein einziges Bit wirklich bekannt. Dieses besagt, ob der Prover das Geheimnis kennt oder nicht.
Protokolle, die die ZK-Eigenschaft besitzen, leiden nicht daran, dass, je öfters diese ausgeführt werden, die Sicherheit weniger wird.
Eine ZK-Technik, welche nicht beweisbar als sicher gilt, ist genauso wenig einsetzbar, wie eine public-key Technik, die als relativ sicher gilt.
Einige weitere Eigenschaften von ZK gegenüber Challenge-Response:

- Viele ZK Techniken vermeiden die explizite Verwendung von Verschlüsselungsalgorithmen.
- Manche auf ZK basierten Techniken sind sehr effizient, wobei manche, die formal die ZK-Eigenschaft besitzen, haben oft einen höheren Overhead (bei Berechnung und/oder Kommunikation) als Public-Key Protokolle, welche die ZK-Eigenschaft nicht haben.
- viele ZK-Protokolle basieren auf den selben unbewiesenen Annahmen, wie auch z.B. Public-Key Techniken auf das Faktorieren oder Problem der Wurzel mod n.

ZK-basiert vs ZK: Viele ZK-Konzepte werden ZK genannt, weil diese die Grundlagen von ZK infolge von "guter" Parameterwahl verwenden. Viele solcher Konzepte, die als ZK-basiert gelten, sind nicht komplett übereinstimmend und werden nicht direkt in praktischen Protokollen eingesetzt. (vgl. [Alf96, S. 405ff])

2.6 Angreifbarkeit bzw. Sicherheitsprobleme von Zero Knowledge

Zero Knowledge ist an sich eine sichere Methode, jedoch gibt es auch hierbei Arten, wie diese Methode missbraucht werden kann: (vgl. [Sch96, S. 109ff])

- **Grand Chessmaster Problem**

  Beim Grand Chessmaster Problem, dargestellt in Abbildung 2.3, geht es darum, dass Alice in zwei verschiedenen Räumen gegen zwei Schachmeister (Gary Kasparov, Anatoly Karpov) spielt, wobei sie einerseits schwarz und im anderen Raum als weißspielt. Beim Spielverlauf beobachtet sie die Züge der Schachmeister und spielt diese Züge beim jeweiligen anderen Gegner. Dadurch wird sie wahrscheinlich gewinnen oder ein Unentschieden erreichen und die Schachmeister würden von Alice's Spielkünsten beeindruckt sein, ohne wirklich zu wissen, dass sie eigentlich gegeneinander und nicht gegen Alice gespielt haben.

  Dieser Betrug kann auch gegen ZK-Identitätsbeweise eingesetzt werden. Hierbei überzeugt Alice Mallory von ihrer Identität, wobei Mallory nun diese Identität (von Alice) Bob beweisen kann.

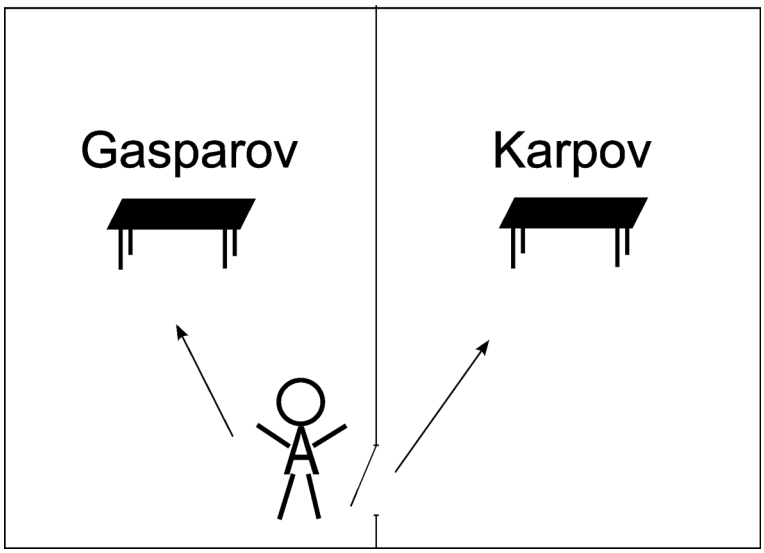

Abbildung 2.3: Grand Chessmaster Problem

- **Mafia Fraud**

  Beim Mafia-Betrug sind vier Akteure beteiligt (Alice, Bob, Carol, Dave). Alice isst in Bob's Restaurant und Carol kauft in Dave's Juweliergeschäft ein. Hierbei sind Bob

und Carol mittels versteckten Mikrofonen bzw. Lautsprechern verbunden.

Wenn nun Alice bezahlen will und ihre Identität an Bob preisgibt, berichtet er Carol davon. Falls nun Dave Fragen stellt, werden diese an Bob weitergeleitet und dieser stellt die Fragen an Alice usw.

Das Resultat daraus ist, dass Dave nun glaubt, dass er die Juwelen an Alice verkauft, jedoch Carol diese mitnimmt.

Das Prinzip dieses Betrugs ist in Abbildung 2.4 dargestellt.

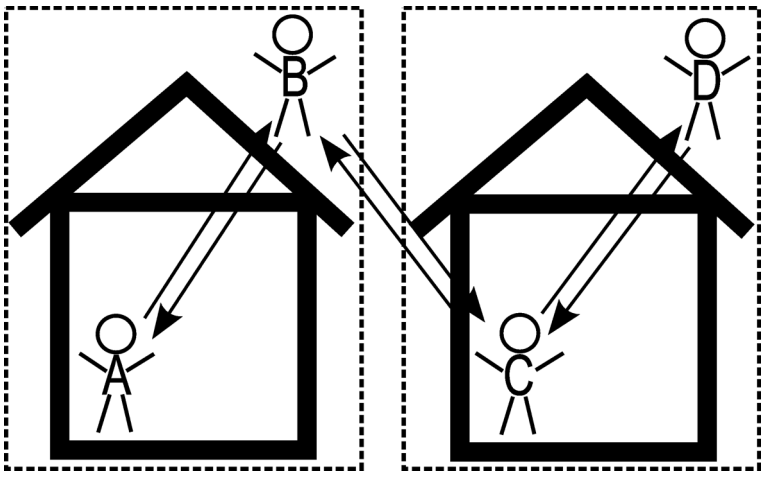

Abbildung 2.4: Mafia-Betrug

- **Terrorist Fraud**

  Carol ist eine bekannte Terroristin und möchte in ein Land kommen. Hierbei kommt ihr Alice zu Hilfe. Alice und Carol sind wieder miteinander verbunden. Dave ist bei der Einwanderungsbehörde angestellt und stellt Fragen (s. Abbildung 2.5)

  Nun kommt Carol zu Dave und möchte einwandern. Dave stellt ihr Fragen, die Carol an Alice weiterleitet. Alice beantwortet diese Fragen und Carol leitet die Antworten an Dave weiter. Dave akzeptiert, weil er denkt, Alice möchte in das Land.

- **Multiple Identity Fraud**

  Bei manchen Implementierungen wird nicht überprüft, wie viele Public-Keys jemand registriert. Daher kann Alice z.B. mehrere Schlüsselpaare besitzen und somit mehrere Identitäten.

- **Renting Passports**

  Alice möchte in ein fremdes Land, bekommt aber kein Visum. Daher verleiht Carol ihre Identität an Alice und kommt somit als Carol in das Land. Hierbei wird aber jedoch nicht die komplette Identität verliehen, sondern nur partielle Informationen.

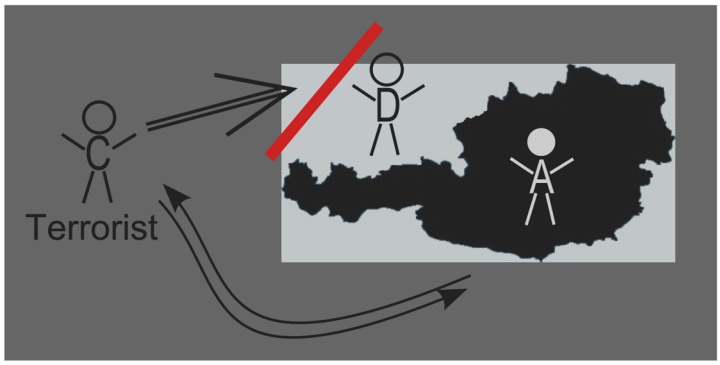

Abbildung 2.5: Terrorist-Betrug

Kapitel 3

Übertragungsmedien

In mobilen Geräten sind viele drahtlose Kommunikationsmittel integriert. Die am Meisten verwendeten sind Infrarot (IrDa), Bluetooth und als neue Technologie NFC (Near Field Communication). Diese sollen betrachtet werden und auf ihre Verwendbarkeit beim Einsatz von Zero Knowledge analysiert werden. Hierbei wird die Betrachtung auch auf die Kombination der Kommunikationsmittel geachtet.
Der Einsatz von Wireless LAN, welches in PDAs (Personal Digital Assistant) und anderen Smartphones bereits eingesetzt ist, wird hier nicht betrachtet, da dies den Rahmen der Diplomarbeit aufgrund der vielen Standards und des weiten Einsatzbereichs sprengen würde.

3.1 Allgemeines

Die hier betrachteten Übertragungsmedien sind

- **IrDa**
  Infrarot galt ursprünglich als Ersatz für Kabelverbindungen und sollte das Arbeiten somit erleichtern. Im laufenden Einsatz und die Spezifizierungen konnten weitere Einsatzzwecke als die "normale" Datenübertragung gefunden werden.

- **Bluetooth**
  Bluetooth ist eine "wirtschaftliche drahtlose Lösung, die komfortabel handhabbar, zuverlässig, einfach zu bedienen und über größere Reichweiten als Infrarot ohne Notwendigkeit direkten Sichtkontaktes einsetzbar ist".[Mul01, S. 31]

- **NFC**

  NFC, eine in Gratkorn/Steiermark entwickelte Lösung, soll den Austausch von Daten über den kontaktlosen Weg ermöglichen.

3.2 Infrarot

Viele Geräte unterstützen die Datenübertragung mittels Infrarot. Bei dieser Datenübertragungsart wird ein Infrarotlicht (Wellenlänge im Bereich von 850-900nm) zwischen den Geräten verwendet. Die Voraussetzung für die Verwendung von Infrarot-Datenübertragung ist eine sehr kurze Entfernung (max. 100cm) und ein direkter Sichtkontakt zwischen den Geräten. Auch funktioniert IrDA nicht durch Wände hindurch. Infrarot ist am ehesten für die Verwendung für Point-to-Point-Verbindungen gedacht, der Hauptverwendungszweck dadurch sind der Austausch von Daten bzw. die Synchronisation mit mobilen Devices/Notebooks etc. Weiters wird IrDA oft bei Fernbedienungen verwendet.
Eine weitere Eigenschaft von IrDA ist, dass dieses nur im Winkel von 30° einsatzfähig ist.
Die *IrDA* (Infrared Data Association) schloss sich 1993 aus ca. 30 Unternehmen (mitunter HP und IBM) zusammen um ein einheitliches Protokoll für die kabellose Datenübertragung zwischen z.B. Drucker und Rechner zu ermöglichen.(vgl. [ird09])
IrDA hat jedoch auch Nachteile. Einer dieser Nachteile ist, dass IrDA im Vergleich zu anderen Funktechnologien nur begrenzte Anwendungen zum Datenaustausch bietet.

3.2.1 Spezifikationen

- IrDA 1.0 - 115,2 kBit/s (Serial Infrared (SIR))
- IrDA 1.1 - 4 MBit/s oder 0,5 MB/s (Mid-Infrared (MIR) (1,152 MBit/s), Fast-Infrared (FIR) (4 MBit/s))
- IrDA 1.3 mit bis zu 16 MBit pro Sekunde (Mid-Infrared (MIR) (1,152 MBit/s), Fast-Infrared (FIR) (4 MBit/s) und Very-Fast-Infrared (VFIR) (16 MBit/s))

3.2.2 Funktionsweise

3.2.2.1 IrDA-Stack

Der IrDA-Stack, der die Grundlage für Infrarot-Datenübertragung bildet, besteht aus den folgenden Schichten(vgl. [Pac98, S. 10ff]):

- IrPHY - physischer Layer
  Im physischen Layer wird das Übertragungssystem, welches auf Basis einer UART-Modulation funktioniert, festgelegt. Weiters werden auch Parameter für die asynchrone serielle halb-duplex-Übertragung festgelegt.

- IrLAP - Infrared Link Access Protocol
  Das IrLAP spezifiziert den auf HDLC (High-Level Data Link Control) basierenden Zugriff auf das Medium und die grundlegende Kommunikation zweier Geräte.

- IrLMP - Infrared Link Management
  Die IrLMP-Schicht liegt über dem IrLAP und hat als grundlegenden Aufgaben das Applikations- und Service Discovery, wobei User Dienste etc. registrieren können (LM-IAS - Link Management Information Access Service) sowie das Multiplexing von Verbindungen über eine IrLAP-Verbindung (LM-MUX).

- TinyTP - Tiny Transport Protocol
  TinyTP als Transportprotokoll implementiert einen Flow-Mechanismus und hat als Hauptaufgaben den Transport von großen Datenmengen (bis zu 4 Gbytes) sowie die Kontrolle des Flusses von Datenpaketen.

Diese Schichten sind auch in Abbildung 3.1 ersichtlich.
Darauf aufbauend werden diverse Kommunikationsprotokolle wie z.B. IrCOMM gelegt, wodurch die darüberliegenden Applikationen (nicht offensichtlich) mit den darunterliegenden Schichten kommunizieren.

3.2.3 Applikationsentwicklung

Mithilfe von IrDA können auf Basis von Infrarot-Technologie Applikationen erstellt werden. Hierzu können APIs für Programmiersprachen bzw. Systeme wie

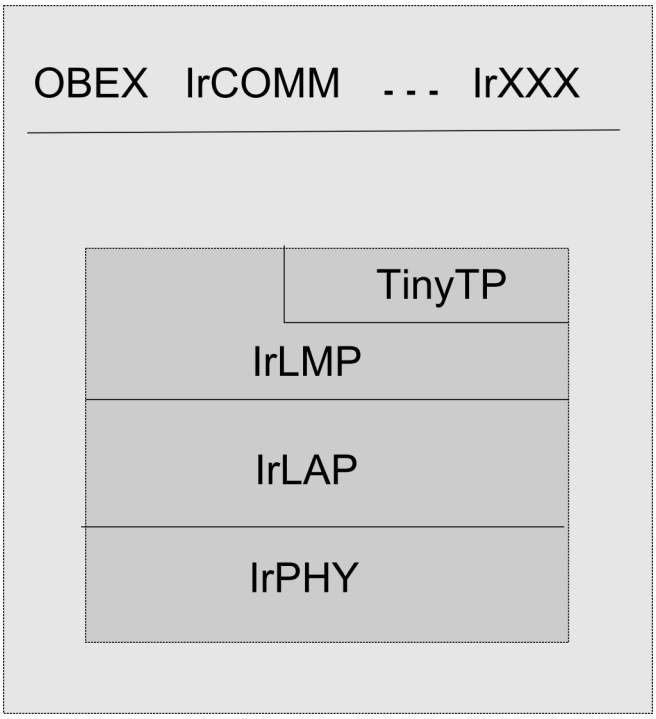

Abbildung 3.1: IrDA-Stack

- C/C++
- Java ([Pac00])
- SymbianOS

verwendet werden und z.B. auf Linux-Systemen entwickelt werden.

3.3 Bluetooth

Bluetooth als Übertragungstechnologie für mobile Geräte wird heutzutage sehr oft verwendet und hat z.B. im Handysegment IrDA abgelöst. Die Anwendungen für Bluetooth sind mit denen von IrDA vergleichbar, jedoch steckt hinter Bluetooth mitunter eine andere Funktionweise bzw. wird mit Bluetooth versucht, einen Nachteil auszubessern.

3.3.1 Entwicklung

1998 wurde der Unternehmensverband *Bluetooth Special Interest Group* (SIG) gegründet. Die Mitglieder der SIG bestehen aus Unternehmen in sehr vielen Bereichen, wodurch die

Entwicklung von Bluetooth begünstigt wurde.
1999 wurde die erste Spezifikation für Bluetooth veröffentlicht. In den darauffolgenden zwei Jahren wurden immer mehr Geräte mit Bluetooth-Funktion vorgestellt. Diese Geräte kamen aus den Bereichen Mobiltelefone, PC-Cards, Headsets, Drucker, Laptops, Eingabegeräte etc. 2001 wurde die Spezifikation in Version 1.1 vorgelegt. 2002 sind mittlerweile 500 Geräte als qualifiziert deklariert. Die IEEE erklärt die Norm IEEE 802.15.1 als Bluetooth-konform. 2003 wird die Spezifikation Version 1.2 eingeführt. Im Jahr darauf (2004) führt die SIG die Spezifikation für Bluetooth 2.0+EDR (Enhanced Data Rate) ein und mittlerweile ist Bluetooth in ca. 250 Mio. Geräten eingebaut. Bis ins Jahr 2007, in dem die Spezifikation 2.1+EDR eingeführt wurde, haben auch z.B. Armbanduhren, Bilderrahmen etc. Bluetooth implementiert. (vgl. [blu09a])

3.3.2 Spezifikationen/Klassen

3.3.2.1 Spezifikationen

Diese Spezifikationen sind in [blu09c] ersichtlich.

- Bluetooth 1.0 bzw. 1.0b
  Erste Spezifikation - hatte Sicherheitsprobleme
  max. Übertragungsrate in Bluetooth 1.x: 732,2 kBit/s
- Bluetooth 1.1
  Signalstärkenindikator hinzugefügt
- Bluetooth 1.2
  durch Einführung von AFH (adaptive frequency hopping) weniger anfällig auf Störungen
- Bluetooth 2.0 + EDR
  durch EDR (Enhanced Data Rate) können Übertragungsraten von bis zu 2,1 MBit/s erreicht werden
- Bluetooth 2.1 + EDR
  Features wie z.B. Quality of Service hinzugefügt

- Seattle Release (Bluetooth 3.0)
  noch nicht verabschiedet. Es sollen Übertragungsraten von bis zu 480 MBit/s mithilfe eines Kanals auf Basis von WLAN bzw. Ultra Wide Band erreicht werden können.

3.3.2.2 Klassen

Die Klassen, in die Bluetooth eingeteilt werden kann, sind in Tabelle 3.1 ersichtlich.

Klasse	Max. Leistung	Reichweite im Freien
Kl. 1	100mW/20dBm	ca. 100m
Kl. 2	2,5mW/4dBm	ca. 10m
Kl. 3	1mW/0dBm	ca. 1m

Tabelle 3.1: Bluetooth-Klassen

3.3.3 Funktionsweise

Bluetooth verwendet das globale und lizenzfreie ISM-Frequenzband bei 2,4 GHz. Dieses Frequenzband (ISM - Industrial, Scientific and Medical) umfasst den Frequenzbereich von 2,4 - 2,484 GHz und benötigt keine Betriebserlaubnis der FCC (Federal Communications Commission). Daher können Bluetooth-Geräte weltweit verwendet werden und mit anderen Bluetooth-Geräten kommunizieren.(vgl. [Mul01, S. 35]
Da Bluetooth nach dem Master-Slave-Prinzip arbeitet, können verschiedene Netzwerkarten verwendet werden (s. Abbildung 3.2). (vgl. [Mul01, S. 41f])

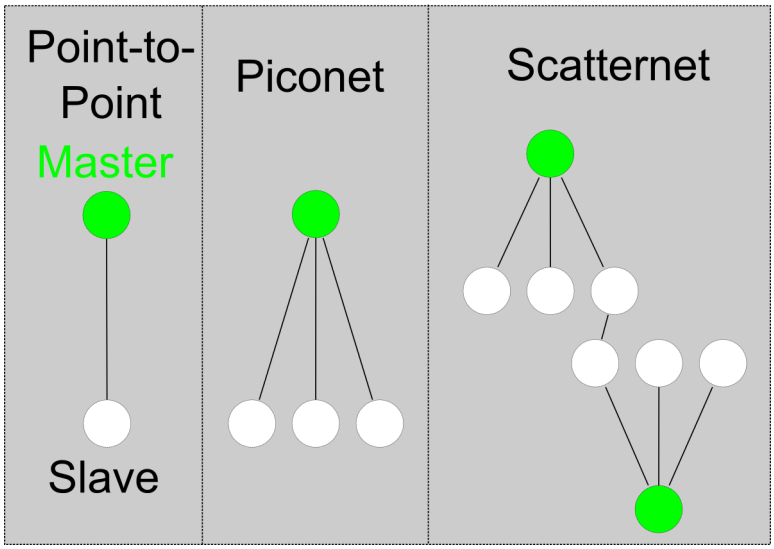

Abbildung 3.2: Bluetooth-Netzwerkarten

Die Bluetooth-Spezifikation definiert zwei Arten von Übertragungsarten: ACL - asynchronous connectionless und SCO - synchronous connection oriented. Bei ACL wird die Datenübertragung nach dem Best-Effort-Prinzip verwaltet, also soviele Daten wie möglich, ohne Acknowledgements etc. Bei SCO wird eine reservierte Bandbreite verwendet und die Verarbeitung erfolgt in Echt-Zeit.
ACL unterstützt symmetrische oder asymmetrische paketbasierende Point-to-Point Verbindungen, die üblicherweise für reinen Datentransfer verwendet werden. SCO bieten symmetrische, circuit-switched, Point-to-Point Verbindungen, die üblicherweise bei Sprach- bzw. Video-Übertragung verwendet wird. Dazu werden drei synchrone 64Kb-Kanäle verwendet, die durch PCM (Pulse Code Modulation) oder CSVD (Continuous Variable Slope Delta) Modulation geteilt werden.(vgl. [Mul01, S. 35f]

Die Bluetooth-Funktion basiert auf einem Protokollstack, der auf Abbildung 3.3 ersichtlich ist.

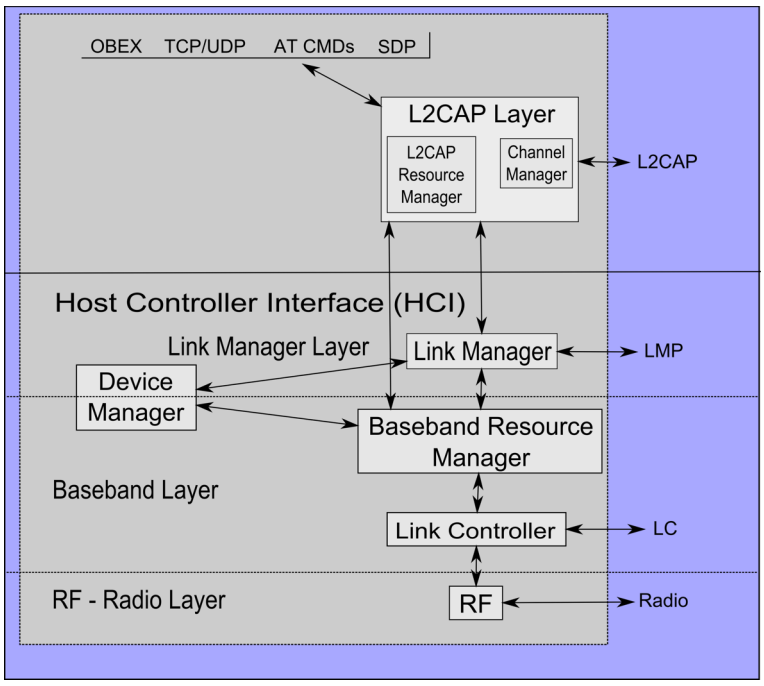

Abbildung 3.3: Bluetooth-Stack

In obiger Abbildung (Abbildung 3.3) sind die wichtigsten Elemente der Bluetooth-Technologie dargestellt (vgl. [Mul01, S. 125f]:

- **Baseband (Basisband)**

  Der Baseband Layer schafft die physikalische Funkverbindung zwischen Bluetooth-Geräten. In dieser Schicht sind Abfrage- und Pagingverfahren inbegriffen, welche die Übertragungsfrequenzen und den Takt zwischen den Bluetooth-Geräten synchronisiert (siehe auch Frequency Hopping, 3.3.4).

  In dieser Schicht befindet sich auch der Link Controller, welcher für Baseband-Protokolle und Low-Level-Verbindungsmanagement zuständig ist.

  Es können in der Baseband-Spezifikation zwei Arten von Datenübertragungsmethoden eingesetzt werden: ACL und SCO (s. 3.3.3).

- **LMP (Link Manager Protocol**

  Die Zuständigkeit des LMP besteht bei der Kontrolle der Verbindungen. So ist das LMP für den Aufbau der Verbindungen, der Steuerung der Verbindung zwischen Bluetooth-Geräten etc. verantwortlich. Weiters wird der LMP für Sicherheitsfeatures (vgl. 3.3.4) eingesetzt, sowie für Energiemodi etc.

  Es ist weiters darauf hinzuweisen, dass LMP-Messages nicht an höherer Schichten weitergeleitet werden und diese im Gegensatz zu Benutzerdaten die höhere Priorität haben. Außerdem müssen LMP-Messages nicht bestätigt werden.

- **L2CAP (Logical Link Control and Adaptation Protocol**

  L2CAP ist das Protokoll, welches z.B. das Multiplexing oder QoS (Quality of Service) der darüberliegenden Schichten übernimmt.

  L2CAP ist, obwohl die Übertragungsarten ACL und SCO definiert wurden, nur für ACL-Traffic zuständig, da SCO-Traffic bzw. -Verbindungen über das Basisband laufen.

- **SDP (Service Discovery Protocol)**

  Das SDP ist die Diensterkennung von Bluetooth. Diese stellt die Grundlage für sehr viele Einsatzmöglichkeiten dar. Durch das SDP können Informationen wie z.B. Dienste, Leistungsmerkmale von Diensten abgefragt werden. Sobald ein Dienst gefunden wurde, kann dieser ausgewählt und über diesen eine Verbindung aufgebaut werden.

Die Bluetooth-SIG hat einige ”usage model” festgelegt, die von verschiedenen Profilen begleitet werden. Hierbei kann man davon ausgehen, dass Geräte, die verschiedene Profile hardwaremäßig unterstützen, ohne Probleme miteinander kommunizieren können.
In einem Profil werden die Meldungen und Prozeduren für die Implementierung einer gewissen Eigenschaft festgelegt, wobei es mandatory und optionale Merkmale gibt. Die verbindli-

chen Merkmale müssen in einer Weise implementiert werden, sodass die Geräte miteinander arbeiten können.
Es gibt vier allgemeine Profile, die für die Datenübertragung verwendet werden(vgl. [blu09c], [Mul01]:

- GAP - Generic Access Profile
  Ist die Grundlage für alle anderen Profile. Bietet die Grundlagen für L2CAP und den Link Manager.
- SPP - Serial Port Profile
  Verwendet, falls Bluetooth als Ersatz für Kabelverbindungen eingesetzt werden soll. Bietet ein Interface für die Verwendung vom RFCOMM-Protokoll
- SDAP - Service Discovery Application Profile
  SDAP stellt das Interface für die Abfrage von Services auf den Bluetooth-Geräten.
- GOEP - Generic Object Exchange Profile
  Das GOEP wird für den Austausch von Objekten verwendet.

Im GAP werden allgemeine Prozeduren zur Erkennung von Bluetooth-Geräten, sowie für das Verbindungsmanagement festgelegt. Dieses Profil beinhaltet, wie in Abbildung 3.4, die weiteren aufgelisteten Profile.

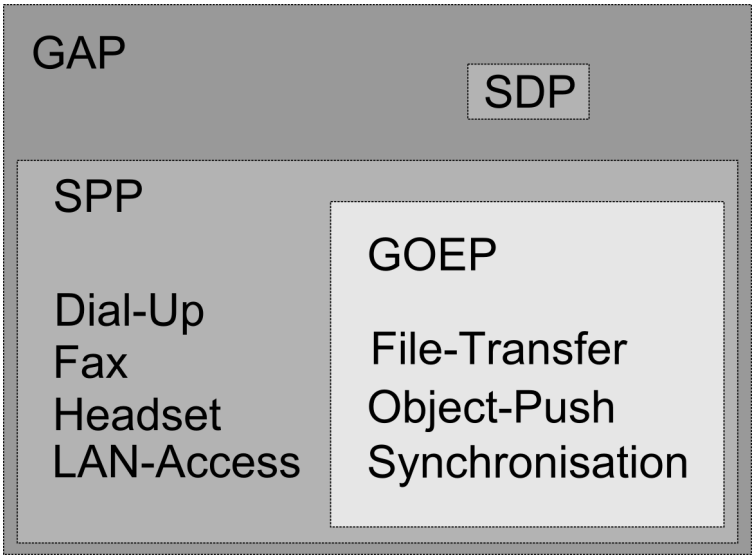

Abbildung 3.4: GAP - Generic Access Profile

3.3.4 Sicherheitsmerkmale

Ein grundsätzliches Sicherheitsmerkmal ist das Frequency Hopping. Bluetooth verwendet eine Spread-Spectrum-Technologie (spezielles Codierverfahren, bei dem das Signal für Andere nur als Rauschen in Erscheinung tritt). Beim Frequency Hopping wechselt der Sender die Frequenzen entsprechend einer zufälligen Sequenz mit einer bestimmten Hopping-Rate (Bluetooth mit z.B. 1600 Hops/s und Frequenzabständen von 1 MHz zwischen 2,402 und 2,480 GHz).(vgl. [Mul01, S. 378]) In der Bluetooth-Spezifikation (s. [blu09c]) sind weitere Sicherheitsmerkmale in der Verbindungsschicht (link level) festgelegt. Diese unterstützen Authentifizierung und Verschlüsselung. Diese Sicherheitsmerkmale basieren auf einem geheimen Verbindungsschlüssel, welcher zwischen zwei Geräten existiert. Um diesen Schlüssel zu erzeugen, wird zwischen den Geräten ein Pairing-Mechanismus bei der ersten Kommunikation durchgeführt.
Im GAP sind folgende Security Modes festgelegt:

- Security Mode 1
  non-secure: keine Sicherheitsmaßnahme wird initiiert
- Security Mode 2
  service-level enforced security: Das Gerät initiiert keine Sicherheitsmaßnahmen bevor der Kanal auf L2CAP-Level steht. Dieser Modus erlaubt es, verschiedene Sicherheitseigenschaften von Applikationen in flexibler Art und Weise zu ermöglichen.
- Security Mode 3
  link level enforced security: Das Gerät initiiert Sicherheitsmaßnahmen bevor die Verbindung steht (auf LMP-Level)

Weiters sieht die Spezifikation zwei Trust-Modi für Geräte vor:

- Trusted Device
  Das Gerät hat eine permanente Verbindung (ist gepaired) und hat Zugriff auf alle Ressourcen und Services
- Untrusted Device
  Das Gerät hat keine permanenten Verbindung, aber möglicherweise eine temporäre bzw hat eine permanente, ist aber nicht trusted. Hierbei hat das Gerät einen restriktiven Zugriff. Unbekannte Geräte gelten auch als untrusted.

3.3.5 Limitationen von Bluetooth

Die folgende Liste deklariert Limitationen von Bluetooth:

- Legacy Applications
  Das Problem hierbei ist, dass diese ”alten” Applikationen den Security Manager nicht kontaktieren, somit muss diese Aufgabe die Hardware erledigen. (vgl. [Mul01, S. 378])
- Device Authentication
  Bei Bluetooth wird ein Gerät authentifiziert, nicht ein User. Daher muss sich eine Applikation selbst darum kümmern, dass die Authentifizierung auf per User-Basis durchgeführt wird.
- Keine Authentifizierung per Service
  Da es keine Authentifizierung auf per Service-Basis gibt, muss für diesen Zweck eine bessere Security Policy implementiert werden, wodurch auch Modifikationen im Security Manager notwendig werden.
- Kein unidirektionaler Traffic
  Da die Übertragungsweise nur zu Beginn der Übertragung festelegt werden kann, ist danach z.B. kein unidirektionaler Traffic möglich.

In der Bluetooth-Spezifikation 2.1 ([blu07, S. 58]) sind gegen MITM bzw. Lauschangriffe (Secure Simple Pairing) zwei Methoden festgelegt:

- Numerischer Vergleich
  und
- Passkey entry

Zusammenfassend lässt sich sagen, dass die Bluetooth-Technologie besonders für Sprach- und Datenanwendung geeignet ist. Es gibt weniger Restriktionen als z.B. bei IrDA in Bezug auf das Durchdringen von Wänden bzw. die Reichweite. Auch kein direkter Sichtkontakt ist vonnöten. Ein weiterer Vorteil sind die Kosten eines Bluetooth-Chips, der laut [blu09b] unter 3 Euro liegen soll.
In Tabelle 3.2 ([Mul01, S. 42]) sind die grundsätzlichen Eigenschaften von Bluetooth noch einmal ersichtlich.

Merkmal	Eigenschaft
Verbindungstyp	Spread Spectrum (mit Frequency Hopping)
Spektrum	ISM-Band mit 2,4 GHz
Übertragungsleistung	1 mW
Datendurchsatz	1 Mbps bei Verwendung von Frequency Hopping
Reichweite	bis zu 10 m
Unterstützte Stationen	bis zu 8 (Pico-Netz)
Sprachkanäle	bis zu 3
Datensicherheit	Authentifizierung: 128 bit Verschlüsselung: 8-128 bit
Adressierung	48-bit MAC-Adresse pro Gerät

Tabelle 3.2: Bluetooth-Eigenschaften

3.3.6 Applikationsentwicklung

Die Entwicklung von Applikationen, welche Bluetooth verwendet, gestaltet sich sehr einfach. Hierzu gibt es z.B. APIs für

- Java - JSR 82 ([Mot08b])
- .NET

Weitere Informationen zur Programmierung von Applikationen, welche Bluetooth verwenden, sind in Kapitel 5 aufgeführt.

3.4 NFC

NFC (Near Field Communication) ist eine Technologie für eine einfache drahtlose Kommunikation zwischen nahegelegenen Geräten. Diese Technologie ist stark in der Entwicklung, und für z.B. mobile Geräte noch wenig erhältlich (Nokia bietet derzeit zwei NFC-fähige Handys an[1][2]), jedoch sollen laut ABI Research ([ABI08]) bis 2011 450 Millionen NFC-fähige Handys am Markt sein.

3.4.1 Entwicklung

NFC wurde 2002 von Philips (bzw. NXP Semiconductors) und Sony entwickelt. Das Ziel war, eine einfach Datenübertragung zwischen zwei Geräten zu ermöglichen. Weiters sollte

[1] http://www.nokia.at/produkte/alle-modelle/nokia-6212-classic
[2] http://www.nokia.at/6131nfc

NFC als Zugriffsberechtigung für z.B. bargeldlose Zahlung, Ticketing etc. verwendet werden können.
2004 wurde das NFC-Forum (s. [nfc09]) gegründet, das die Entwicklung und Verbreitung von NFC zum Ziel hat. Laut [nfc09] sind die Ziele:

- "Develop standards-based Near Field Communication specifications that define a modular architecture and interoperability parameters for NFC devices and protocols
- Encourage the development of products using NFC Forum specifications
- Work to ensure that products claiming NFC capabilities comply with NFC Forum specifications
- Educate consumers and enterprises globally about NFC"

2005 werden die ersten Usability Tests durchgeführt und das erste kommerzielle Telefon war erhältlich. 2006 waren bereits mehr als 100 Mitglieder des Forums, darunter unter anderem HP, Toshiba, Visa etc. Seit 2006 laufen auch einige Feldversuche für den Einsatz und die Verbreitung von NFC. Im September 2007 startete z.B. die Mobilkom Austria das berührungslose Bezahlen von Tickets der ÖBB bzw. der Wiener Linien (s. [Aus09]).

3.4.2 Standards

NFC wurde standardisiert durch die Standards

- ISO 18092 ([Int04])
- ECMA 340, 352, 356, 362, 373, 385, 386 ([ECM08])
- ETSI TS 102 190 ([ets03])

und ist kompatibel mit der ISO 14443-Norm ([Int08]) betreffend kontaktlosen Karten und Kartenlesern.

3.4.3 Funktionsweise

NFC arbeitet durch Magnetfeldinduktion (aktiv) bzw. durch Lastmodulation (passiv) auf kurzen Distanzen (ca. 10cm) und auf der Frequenz von 13,56 MHz. NFC arbeitet mit Über-

tragungsraten von 106 kBit/s, 212 kBit/s oder 424 kBit/s.
NFC kann in 2 Modi betrieben werden: aktiv und passiv (vgl. [ets03, S. 12]).

- aktiv

  Beim aktiven Modus kommunizieren sowohl der Initiator als auch der Kommunikationspartner über jeweils ein eigenes aufgebautes Feld, welches abwechselnd aufgebaut wird.

- passiv

  Beim passiven Modus baut das initiierende Gerät das Feld auf und stellt dieses dem Partner zur Verfügung, der somit die Energie, die benötigt wird, von diesem elektromagnetischem Feld abnimmt und als Transponder fungiert. Der Partner moduliert das existierende Feld um die Nachrichten an den Initiator zu senden.

Die Übertragungsraten und der dazugehörige Modus sind in Abbildung 3.5(in Anlehnung an [ets03, Table 1, S.13]) ersichtlich. Bei NFC werden je nach Übertragungsart verschiedene

Communication Mode	kbps
active or passive	106
active or passive	212
active or passive	424
Active	848
Active	1 667
Active	3 390
Active	6 670

Abbildung 3.5: NFC Modi und dazugehörige Übertragungsraten

Kodierungsverfahren angewendet. Bei 106kBd wird eine modifizierte Miller-Kodierung verwendet, darüber hinaus wird die Manchester-Kodierung benutzt.
NFC bietet unter anderem dem Vorteil, dass Daten gleichzeitig gesendet und empfangen werden können. So kann auf einfache Weise eine Kollision entdeckt werden.

3.4.3.1 General Protocol flow

Der grundsätzliche Protokollablauf (Geräterkennung) zwischen NFC-Geräten sollte wie folgt ausgeführt werden (vgl. [ets03, S.19]):

- Jedes NFC-Gerät sollte sich per Default im Target Mode befinden

- Im Target Mode sollte das Gerät kein Feld erzeugen und auf einen Befehl vom Initiator warten.
- Das NFC-Gerät darf nur in den Initiator mode schalten, wenn es von der Applikation benötigt wird.
- Die Applikation sollte den Betriebsmodus und die Übertragungsgeschwindigkeit entscheiden.
- Der Initiator sollte überprüfen, ob ein externes Feld vorhanden ist, und falls ja, kein eigenes erzeugen.
- Wenn kein externes Feld gefunden wurde, sollte das eigene Feld aktiviert werden.
- Das Zielgerät sollte durch das Feld des Initiators aktiviert werden.
- Ein Befehl vom Inititator wird entweder im aktiven oder im passivem Modus mit gewählter Übertragungsgeschwindigkeit übertragen
- Die Übertragung einer Response vom Zielgerät wird entweder im aktiven oder im passiven Modus durchgeführt. Der Modus und die Übertragungsgeschwindigkeit sollte derselbe sein, wie beim Initiator.

Der Ablauf ist auch in Abbildung 3.6(s. [ets03, S.20]) ersichtlich.

3.4.3.2 Angriffsmöglichkeiten (vgl. [Ern06, S. 4ff])

NFC ist grundsätzlich nicht sicher vor Lauschangriffen. Dies soll zwar durch die kleine Entfernung ausgeglichen werden, d.h. jemand der mithören will, muss sehr nahe an dem Betroffenen sein bzw. kann das Signal mithilfe einer Antenne empfangen. Die Distanz hängt dabei von z.B. der Umgebung, der Antenne etc. ab.
Auch können die Daten durch eine Einspeisung, welche gut koordiniert stattfinden und im richtigen Frequenzspektrum platziert sein muss, zerstört werden. D.h. dies ist mehr oder weniger ein DoS (Denial of Service)-Angriff.
Weiters könnte der Angreifer die Daten manipulieren wollen, d.h. geänderte Daten senden. Dieser Angriff ist etwas komplizierter, weil dabei kontrolliert in die Modulationsverfahren eingegriffen werden muss.
Falls Daten eingefügt werden sollen, muss dies so stattfinden, dass diese gesendet werden,

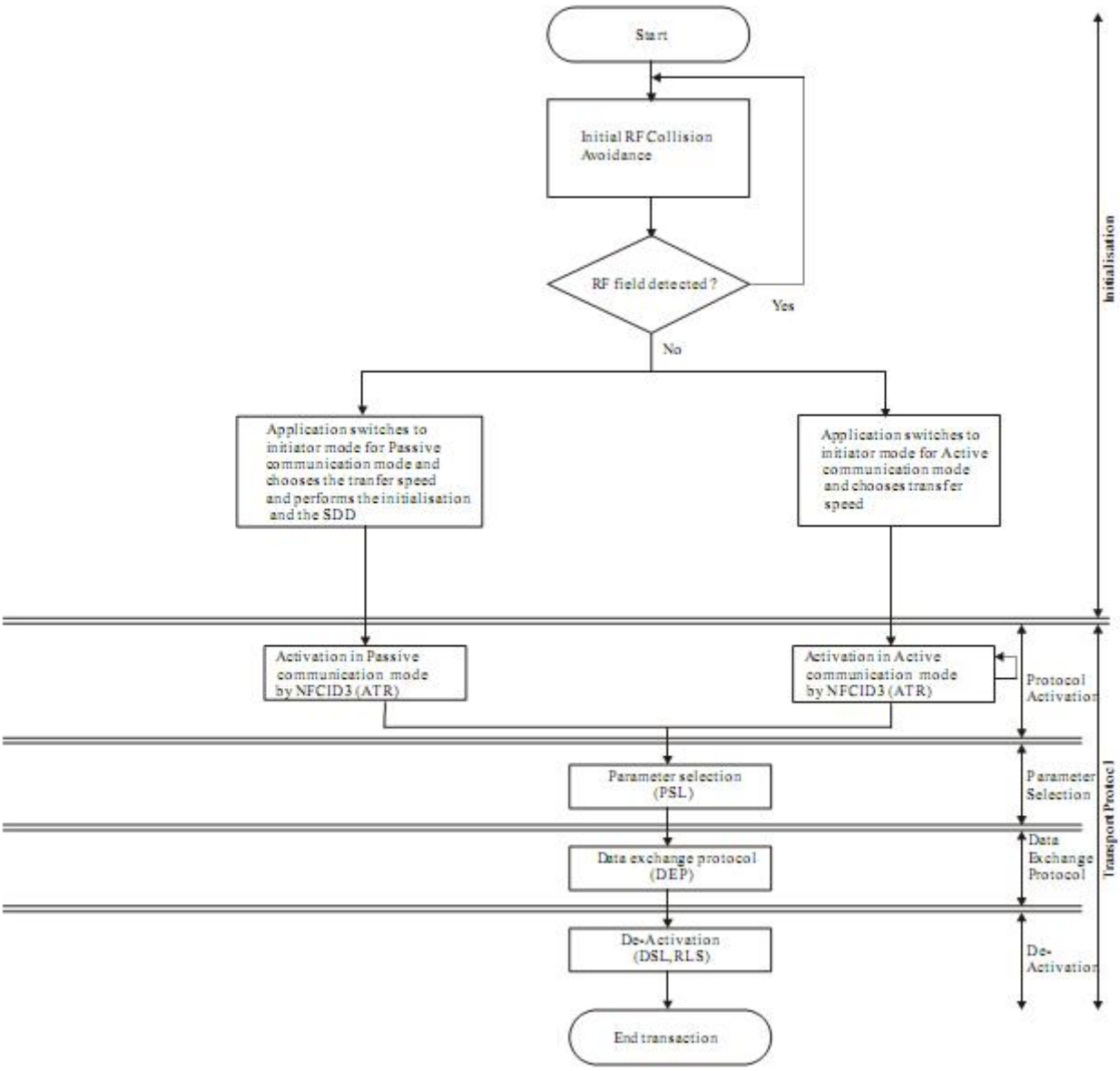

Abbildung 3.6: NFC Protokollablauf

bevor das zweite Device antwortet.

Eine MITM-Attacke ist bei NFC praktisch unmöglich, da durch die Kommunikationsmodi dies restriktiert wird.

3.4.3.3 Einsatz

Der Einsatz von NFC soll zum größten Teil in mobilen Geräten erfolgen.

Einsatzzwecke hierbei sind z.B. mobile Ticketing, mobile payment, mobile commerce etc.

Das NFC-Forum beschreibt dies in der Form

"The possibilities are endless, and NFC is sure to take the complexities out of today's increasingly sophisticated consumer devices and make them simpler to use."[nfc]

Eine weitere interessante Möglichkeit ist, dass ein Pairing-Vorgang zwischen Bluetooth (2.1[blu07])-Geräten mit NFC stattfinden kann. Hierbei wird Bluetooth durch einfaches Berühren der mobilen Geräten aktiviert. (vgl. [Tao06])

3.4.4 Applikationsentwicklung

Für die Entwicklung von Applikationen mit NFC-Support ist auch ein JSR vorhanden, womit Applikationen auf Java-Basis entwickelt werden können. Hierzu werden die APIs von JSR 275 ([Nok06]) verwendet.

3.5 Bewertung der verschiedenen Übertragungsmedien

Die beschriebenen Übertragungsmedien haben jede verschiedene Eigenschaften. Diese sind in Tabelle 3.3 kurz zusammengefasst.

Parameter	**NFC**	**Bluetooth**	**IrDa**
Netzwerk Konfiguration	Peer-to-Peer	Point-to-Multipoint	Point-to-Point
Reichweite	0.1 m	10 m (bis zu 100 m)	1 m
Frequenz	13,56 MHz	2,4 GHz	
Geschwindigkeit	bis zu 424 kbit/s	bis zu 721 kbit/s	115 kbit/s (SIR)
Setup-Zeit	< 0.1 s	etwa 6 s	0.5 s
Sicherheit	auf Hardware- und Protokollebene	auf Protokollebene	
Kommunikations-Modi	aktiv-passive, aktiv-aktiv	aktik-aktiv	aktiv-aktiv
Applikationen	Verbindungen, Datenaustausch, RFID, Bezahlung	Datenaustausch, Headsets	
Infrastruktur	kontaktlose RFID-Lösungen	mobile Geräte	
Usability	einfach, sachdienlich, touch and go	nicht bequem aufgrund der langen Setup-Zeit	
Kosten	niedrig	moderat	niedrig
RFID-Kompatibilität	ja	nein	nein

Tabelle 3.3: Vergleich der Technologien (in Anlehnung an [Tao06])

3.5.0.1 IrDA

+ günstig
+ effizient
+ zuverlässig
+ störunanfällig
+ geringer Stromverbrauch
+/- kurze Entfernung
- Applikationen
- mittlerweile wieder Rückgang bei Handys
- Richtungsabhängigkeit
- Sichtlinie zwischen den Geräten

3.5.0.2 Bluetooth

+ keine Lizenzgebühren
+ Verwendung in nahezu alle Geräten, inkl Haushaltsgeräte
+ kein Sichtkontakt
+ Applikationen
+/- mehrere Geräte zu einer Gruppe zusammengefasst
- Gerät muss ausgewählt werden
- Sicherheit
- eindeutige Geräteadresse

3.5.0.3 NFC

+ Einfacher Anwendungsablauf
+ Möglichkeit der Kombination mit Bluetooth
+ kompatibel zu ISO 14443 (Smartcards)
+ schneller Verbindungsaufbau
+ Energieeffizient
+ vielseitige Verwendung
+ einfache Konfiguration
+/- geringe Reichweite

- nur zwei Geräte kommunizieren miteinander
- Übertragungsrate

3.5.1 Fazit

Die betrachteten Übertragungsmöglichkeiten haben allesamt ihre Vor- und Nachteile. Für den Einsatz von Zero Knowledge jedoch werden Faktoren wie z.B. einfache Implementierung bzw. einfache Anpassung der Parameter wichtig.
Da hierbei der Faktor Implementierung eine sehr große Rolle spielt, sind die besten Übertragungsarten zweifellos Bluetooth und NFC.
Welche nun wirklich zum Einsatz kommt, basiert auf den wirklich Einsatzzweck und die Umgebung. Falls z.B. eine Authentifizierung auf kontaktlose Art und schnell von statten gehen soll, ist NFC die bessere Lösung.
Aufgrund der Tatsache, dass die Bluetooth-APIs für Java bereits länger existieren und kein NFC-Device verfügbar war, wurde bei der praktischen Forschungsstellung Bluetooth als Übetragungsweg gewählt.

Kapitel 4

Einsatzmöglichkeiten von Zero Knowledge

In diesem Kapitel werden grundsätzlich die Einsatzmöglichkeiten von ZK-Protokollen bzw. ZK konkret behandelt.
Hierzu werden verschiedene Einsatzbereiche betrachtet, die in Kategorien eingeteilt werden. Diese Kategorien sind u.a.

- Unternehmenseinsatz bzw. Einsatz in Unternehmen
- Einsatz als zusätzliches integriertes Feature
  und als Nebenkategorie bzw. weniger behandelt
- Privateinsatz

Diese Kategorien wurden gewählt, da der Einsatz von Zero Knowledge meist nicht offensichtlich ist und deshalb für den Privateinsatz wenig Relevanz besitzt bzw. für den Enduser (mehr oder weniger) nicht ersichtlich ist.

4.1 WLAN-Authentifizierung

Das Thema der Authentifizierung bei WLAN (Wireless LAN) ist durch die weite Verbreitung von WLAN immer elementarer geworden.

Viele Geräte bzw. Hersteller haben eine MAC-Adressen-(Media Access Control-Adressen-)Zugriffsrestriktion (MAC-Filter) implementiert, welcher jedoch durch MAC-Spoofing (fälschen der MAC-Adresse) einfach umgangen werden kann. Eine Möglichkeit dies zu Verhindern ist der Einsatz von einem IDS (Intrusion Detection System), welches dieses erkennen kann (falls z.B. zwei gleiche MAC-Adressen im Netzwerk vorhanden sind wird die Verbindung für beide getrennt).
Die aktuell implementierten Verschlüsselungsmethoden (wie z.B. WEP(Wired Equivalent Privacy), WPA (Wi-Fi Protected Access) bzw. WPA2) besitzen integrierte Authentifikationsmethoden. (vgl. IEEE 802.11 [IEE])
Diese sind z.B.:

- Open System - WEP

  Hierbei findet keine Authentifizierung statt (wenn keine Verschlüsselung konfiguriert ist). Dies ist die unsicherste Variante, wobei sich jeder Client verbinden kann.

  Ist WEP als Verschlüsselung konfiguriert wird der WEP-Key zur Authentifizierung verwendet. Der Client ist somit authentifiziert, falls jedoch der WEP-Key nicht übereinstimmt, kann keine Kommunikation stattfinden.

- Shared Key - WEP

  Bei WEP wird bei der Authentifizierung mittels Shared Key-Authentication diese über Challenge-Response durchgeführt.

  Hinweis: Es ist bewiesen, dass WEP unsicher ist und die WEP-Verschlüsselung gebrochen werden kann, daher ist die Verwendung von WEP nicht empfehlbar bzw. praktisch irrelevant.

- Preshared Key - WPA/WPA2

  Bei WPA-PSK (WPA-Preshared Key) bzw. WPA2-PSK wird die Authentifizierung mittels einem Preshared Key durchgeführt. Dieser ist ein Passwort welches 8-63 Zeichen lang sein kann.

  Das Problem beim Preshared Key ist vergleichbar zu einem normalen Passwort die Länge der gewählten Phrase.

- IEEE 802.1X

  Der IEE 802.1X-Standard ist Standard zur Authentifizierung in Netzwerken.

  Hierbei wird ein Client auf einem sogenannten Authenticator authentifiziert. Dieser überprüft die Authentizität des Clients (mithilfe einer Nachfrage der User-Credentials

am Authentication Server (z.B. RADIUS). Der Authenticator weist dann die Anfrage zurück oder gibt den Zugriff auf die gewünschten Ressourcen frei.

RADIUS (vgl. Imprementierung FreeRADIUS [fre09]) ist ein Netzwerkprotokoll, welches in diversen RFCs definiert wurde (Basis: RFC2058 und RFC2059[1]).

An diese (Authentifizierungs-)Methoden kann ZK ansetzen.
Zero Knowledge kann bei der Problematik der MAC-Filterliste eine Abhilfe schaffen, in dem der Zugriff nur Geräten erlaubt wird, welche das Geheimnis kennen. Hierbei wird jedoch ein wichtiger Faktor sein, wie das Geheimnis an die Berechtigten Teilnehmer "verteilt" wird. Eine Möglichkeit besteht darin, dass dieses in der Hardware integriert ist, was den Nachteil mit sich bringt, dass dieses (theoretisch) nicht geändert werden kann, wenn der Bedarf einer Änderung besteht.
Eine weitere Möglichkeit besteht darin, dies per Software zu lösen, was aber einen Risikofaktor mit sich bringt, da jede Software decompiliert (vom Binärcode in den Sourcecode umgewandelt) werden kann. Eine Abhilfe gegen die Decompilierung kann ein integrierter Verschlüsselungsmechanismus sein, welcher den Binärcode schützt.
Auch im Bereich der WLAN-Protokolle kann Zero Knowledge eingesetzt werden. Wie oben ersichtlich, werden bei den Standard-Protokollen verschiedene Authentifizierungsmethoden verwendet, die z.B. auf Shared-Key-Verfahren auf Challenge-Response-Basis aufbauen. Hierzu kann nun ein Zero Knowledge-Protokoll ersatzweise eingesetzt werden, wobei wiederum die Lösung hardware- oder softwarebasiert sein kann. Die softwarebasierte Lösung könnte zusätzlich eine weitere Authentifizierungsmethode, z.B. Biometrie verwenden, um die ZK-Prozedur "freizuschalten" .

4.2 Authentifizierung an einem Server

Die ZK-Technologie kann z.B. auch ein Authentifizierungsmechanismus zur Anmeldung an einen Server sein.
Hierbei spielt die Art des Servers (Web-Server, Domain-Server etc.) keine Rolle, da ZK als Protokoll über bzw. unter der "Standard-Authentifizierungsschicht" liegt. Falls im Unternehmenseinsatz LDAP (Lightweight Directory Access Protocol), wird ZK nicht unbedingt als

[1] http://tools.ietf.org/html/rfc2058 bzw http://tools.ietf.org/html/rfc2059

Ablösung der LDAP-Authentication dienen. Als zusätzliche Authentifizierung jedoch könnte ZK verwendet werden.

Falls jedoch ein kleiners, weniger komplexes System eingesetzt wird (z.B. bei SOHO (Small Office, Home Office)), bei dem Services durch eine andere Authentifizierungsmethode veröffentlicht werden, kann ZK durchaus eine Alternative bieten.

Zero Knowledge-Authentifizierung muss jedoch eigens implementiert werden, was bedeutet, dass dadurch im Vorfeld ein Mehraufwand einzurechnen ist, welcher bei bereits implementierten bzw. etablierten Methoden durch Installationsroutinen wegfällt.

4.3 ZK als Authentifizierungsschicht innerhalb von Protokollen

Die grundsätzliche Einsatzvariante von Zero Knowledge ist die Verwendung als unterliegende Authentifizierungsschicht.

Für jedwegliche Anwendungen soll Zero Knowledge die Authentifizierung übernehmen, d.h. Dienste, die über oder unterhalb der Authentifizierungsschicht liegen, leiten den Authentifizierungsaufrag an das ZK-Protokoll weiter und bekommen Rückmeldung, ob die Authentifizierung erfolgreich war oder nicht. Basierend darauf ermöglichen oder verweigern diese ihren Dienst. Abbildung 4.1 zeigt dies beispielhaft. Da die Authentifizierungsschicht als ei-

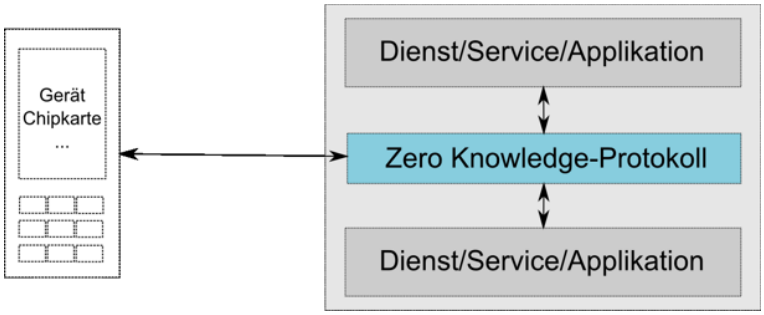

Abbildung 4.1: Zero Knowledge als Authentifikationsschicht

gene Schicht dient und somit abgeschlossen agieren kann, müssen sich die Service-Schichten nicht um die Authentifizierung kümmern und bekommen nur den Return-Wert der Authentifikation zur Verfügung gestellt.

4.4 Geschützte Inhalte

4.4.1 Offline

Zero Knowledge kann bei Offline-Inhalten Einsatz finden. Hierbei gibt es die Möglichkeit, ZK beim DRM (Digital Rights Management) einzusetzen, welches das Nutzungsrecht von digitalen Daten (wie z.B. Software, Musik etc) steuern soll.
Diese Maßnahme kann möglicherweise an Hardware gebunden sein, wobei auf die Thematiken Kopierschutz bzw. auf Dongles (s. 4.6) verwiesen wird.

4.4.2 Online

Die Idee zum Einsatz von Zero Knowledge bei Online-Inhalten bezieht sich auf die Verwertbarkeit bzw. Verwendbarkeit im Bereich Medien aller Art, von Musik über Videos bis hin zu Spielen oder Kommunikationsnetzwerken.
Hierbei wird der Blick Richtung Mediennutzung bzw. Angebote oder Nutzung der Inhalte per Mobiltelefon gerichtet. Da diese Geräte mittlerweile viele Features eingebaut haben, welche die Nutzung dieser Medieninhalte ermöglichen (z.B. DVB-H, Internetzugang etc), kann hier abhängig vom Betreiber eine Authentifizierung (im Sinne von Zugriffsberechtigung) über Zero Knowledge erfolgen. Die Authentifizierung findet über z.B. GSM oder UMTS statt, wobei die Übertragung verschlüsselt vonstatten gehen muss.
Dadurch ergeben sich Möglichkeiten, dass die Authentifizierung über das Gerät stattfindet (bzw. über die SIM-Karte) und nicht über Eingabe von Benutzerdaten.

4.5 Authentifizierung im Web

Der Einsatz von ZK für die Authentifizierung im Web ist eingeschränkt, bzw. das aufwendigste Thema ist die Verteilung des Geheimnisses, also wer dieses Geheimnis besitzen darf. Um ZK als Authentifikationstechnologie der Wahl einzusetzen, müssen also verschiedene Parameter gewählt werden, die außerhalb des Rahmens dieser Arbeit liegen.
Grundsätzlich kann gesagt werden, dass ZK allein durch den (noch so geringen) Implementierungsaufwand für den Einsatz speziell konfiguriert und implementiert werden muss. Weiters sind bereits andere Authentifikationsmethoden (wie z.B. die Windows-Authentication, wel-

che bereits in IIS[2] integriert ist), einsatzfähig und können in einfachster Weise verwendet werden.
Hierbei wird jedoch wieder auf 4.2 verwiesen, dass im SOHO-Bereich, falls keine speziellen Mechanismen zur Authentifizierung verwendet werden, ZK als Alternative in Frage kommen kann.

4.6 Dongles

Ein Dongle ist ein Gerät, meist in Form eines USB-Sticks, welches dazu dient, Software vor unerlaubter Vervielfältigung zu schützen. Hierbei kann z.B. eine Applikation überprüfen, ob ein Dongle vorhanden ist und falls ja, vorhandene Programmfunktionen freigegeben bzw. diese bei Nicht-Vorhandensein deaktivieren.
Hierbei kann durch ZK eine zusätzliche Authentifizierungs-Schicht eingepflegt werden, wobei das Geheimnis z.B. die Lizenzart der Applikation ist (d.h. jedoch in diesem Beispiel auch, dass je nach Lizenzanzahl der Vorgang öfters wiederholt werden muss). Die Software fragt den Dongle ab und "berprüft das Geheimnis und kann somit die Applikation freischalten.

4.7 Token

Token bzw. Security Token stellen ein Hardware-Modul dar, welche die Authentifizierung an verschiedenen Systemen übernehmen soll. Diese Module können in Form von z.B. Smartcards, USB-Sticks, RFID-Transponder, Mikrochips etc. eingesetzt werden.
Die Authentifizierung erfolgt durch eine Challenge-Response-Authentifizierung, wobei evtl. weitere andere Maßnahmen, wie z.B. Biometrie, zusätzlich eingesetzt werden.
Der Vorteil von Tokens liegt darin, dass bei der Verwendung von Tokens, wie bereits erwähnt, eine weitere Maßnahme benutzt wird. Dies kann eine PIN sein, ein Fingerabdruck etc. Ein weiterer Vorteil ist, dass ein Token einmalig ist und nicht vervielfältigt werden kann und falls dieser verloren wurde, durch das System gesperrt werden kann.
Wie bereits erwähnt, wird bei der Token-Authentifizierung Challenge-Response verwendet. Hier kann ZK anschließen. ZK kann anstelle der Challenge-Response-Verfahrens eingesetzt werden, um den Authentifizierungsvorgang durchzuführen.

[2] http://www.iis.net/

4.8 weitere Möglichkeiten

- Pay-TV

  Es ist möglich, Zero Knowledge-Protokolle (wie z.B. Fiat-Shamir) auf Chipkarten zu implementieren. Dadurch haben diese einen wesentlichen Vorteil gegenüber manchen Public-Key-Verfahren.

  Laut [Alb06, S. 47] wurde das Fiat-Shamir-Verfahren (s. 2.4.2) bereits 1991 verwendet und in einer Patentanmeldung für Pay-TV (bekannt als "Videocrypt") dargestellt.

- Container für verschlüsselte Daten

  Es wird immer wichtiger, persönliche oder vertrauenswürdige Daten an einem sicheren Ort aufzubewahren bzw. zu speichern.

  Hierzu kann mithilfe einer Verschlüsselung und ZK als Authentifizierungsmechanismus eine sichere Variante gestellt werden (vgl. 4.7).

  Als als Speicher von PGP (GPG, etc)-Schlüsseln kann ZK als Freigabemechanismus fungieren. Weiters können ZK-Protokolle auch als digitale Signatur eingesetzt werden bzw. steht die Signierung an einer vorderen Position.

- Zutrittsberechtigung

  Auch für den Einsatz als Zutrittsberechtigung(zu Gebäuden, KFZ etc.) kann ein ZK-Verfahren verwendet werden, wobei hierbei wieder zum Einsatz als Token verwiesen wird.

- SIM-Karte

  Die Authentifizierung an einem Mobilbetreiber-Netz kann mittels ZK erfolgen. Da ZK-Protokolle auch mitunter auf Chipkarten (hier z.B. SIM - Subscriber Identity Module) eingesetzt werden können bzw. durch ihre Eigenschaften prädestiniert dafür sind.

- als Zahlungsmittel

  Auch ZK in z.B. Kombination mit NFC kann gemeinsamen Einsatz als Zahlungsmittel Verwendung finden. Hierbei kann der Einsatz beim Fahrkartenkauf, Snackautomaten etc., wie beim Pilotprojekt der Mobilkom Austria (s. [Aus09]) stattfinden.

4.9 Praktikabilität von Zero Knowledge

Die Tatsache, dass Zero Knowledge ein modifiziertes Challenge-Response-Verfahren ist, impliziert, dass dieses den selben Einsatzbereich haben kann wie Challenge-Response. Da Zero Knowledge im Vergleich zu Challenge-Response aufgrund der kleinen Berechnungszeiten und des einfachen Schemas auf Chipkarten eingesetzt werden, wodurch ein breites Spektrum an Anwendungsmöglichkeiten ersichtlich wird.

Beim praktischen Einsatz kommt es jedoch immer darauf an, welche Applikationen bzw. welches Ziel angestrebt wird, um das Authentifizierungsverfahren darauf anzupassen. Wie bereits erwähnt, sind die Zero Knowledge-Protokolle nur ein Grundsatz und können durch kluge Parameterwahl bzw. Modifikationen des Protokolls soweit abgeändert werden, dass dieses dem gewünschten System entspricht.

Ein gravierender Faktor beim Einsatz von Zero Knowledge ist die Verteilung des Geheimnisses, welches durch den Prover bewiesen werden soll. Im Unternehmenseinsatz kann dies durch eine Unternehmenspolicy geregelt sein (Hard- oder Software). Beim Einsatz in Software kann dies auch mithilfe eines Secrecy-Servers (der die Geheimnisse an die Clients verteilt) durchgeführt werden, wobei natürlich hierbei auch auf die Authentifizierung auf diesem Wert gelegt werden muss.

Zusammengefasst ist der Einsatz von Zero Knowledge durchaus praktikabel, jedoch muss die Entwicklung bzw. die Forschung für den praktischen Einsatz noch fokussierter stattfinden. Es gibt Lösungsansätze für die Anwendung, jedoch sind diese spezifisch (z.B. für Unternehmen) und nicht für dein Einsatz in der breiten Masse tauglich.

Kapitel 5

Einfache Forschungsstellung

Diese Forschungsstellung soll zeigen, wie in einer relativ simplen Weise ein Zero Knowledge-Protokoll implementiert werden kann.
Hierbei wird mittels der Bluetooth-Technologie, welcher heutzutage Standard bei mobilen Geräten ist, ein Zero Knowledge-Authentifizierungsvorgang durchgeführt.

5.1 Bedingungen

Folgende Bedingungen werden für die Forschungsstellung gesetzt:

- ZK-Protokoll

  Der verwendete Algorithmus basiert auf dem Fiat-Shamir-Protokoll (s. 2.4.2), da der Sinn dieser Forschungsstellung nicht die Darstellung bzw. Implementierung der jeweils gewählten Protokolle beinhaltet, sondern nur als Exempel statuiert ist.

- Verwendetes Equipment:

  - Geräte

    Zur Entwicklung bzw. für die durchgeführten Testläufe wurden folgende Geräte verwendet:

    * Nokia 6288[1]
    * Nokia 3109c[2]

[1] http://www.nokia.at/link?cid=PLAIN_TEXT_53549, 2009-02-04
[2] http://www.nokia.at/link?cid=PLAIN_TEXT_217097, 2009-02-04

* Samsung SGH-D900i[3]

Alle diese Geräte unterstützen den JSR-82 (BTAPI - [Mot08b]).

– Software

Die gewählte Entwicklungsumgebung für diese Forschungsstellung war die NetBeans IDE[4] in der Version 6.5, wobei die gewählte Version bereits alle Voraussetzungen für die Entwicklung von J2ME-Applikationen beinhaltet.

– Programmierung

Die gewählte Programmiersprache ist Java 2 Micro Edition, da mittlerweile sogut wie jedes Mobiltelefon Java-fähig ist. Wie bereits erwähnt ist hierbei jedoch wichtig, dass das Mobiltelefon den JSR-82 unterstützt, um die benötigte Bluetooth-Funktionalität zu gewährleisten.

5.1.1 Grundlagen für die Programmierung

5.1.1.1 J2ME

Die Grundlage für die Programmierung dieser Forschungsstellung bildet die Java 2 Micro Edition.
Laut Sun (s. [Sun09]) ist J2ME

"[...] a collection of technologies and specifications to create a platform that fits the requirements for mobile devices such as consumer products, embedded devices, and advanced mobile devices. It is a collection of technologies and specifications that can be combined to create a complete Java runtime environment specifically to fit the requirements of a particular device or market."

Diese Technologie zielt darauf ab, die Grundfunktionalität von Java auf eine große Masse von Geräten zu platzieren und somit die Interoperabilität zwischen vielen (bzw. allen) Geräten und Herstellern zu garantieren. Je nach Hersteller werden verschiedene Teile speziell unterstützt. Hierbei spielt der JSR-82 (BTAPI - Bluetooth API) eine große Rolle.
Der JSR-82 (Java Specification Request-82) dient zur Spezifikation der Programmierschnittstelle zur Verwendung von Bluetooth in Applikationen.

[3] http://telecom.samsung.de/produkte/detail11_main.aspx?guid=1b134e3c-c49a-4568-af4c-f88b19c98181, 2009-02-04
[4] http://www.netbeans.org

Die Java Micro Edition wird grundsätzlich in zwei Bereiche unterteilt:

- CLDC

  Die CLDC (Connected Limited Device Configuration) ist die ”Basisversion”, welche auf leistungsschwächeren Geräten eingesetzt wird. Bei dieser Forschungsstellung reicht diese Configuration in der Version 1.0.

- CDC

  Das CDC (Connected Device Profile) ist die leistungsumfassendere Variante und mit dieser können unter anderem Applikationen für High-End PDAs, Set-Top Boxen etc. erstellt werden.

Oft wird die CLDC mit dem MIDP (Mobile Information Device Profile) kombiniert, welches die am Meisten implementierte Variante auf z.B. Mobiltelefonen darstellt. Diese Kombination wird auch hierbei verwendet.
Für diese Kombination wird ein MIDlet (Applikation) erstellt, welches auf dem Gerät installiert wird.
In Abbildung 5.1 wird die Struktur von CLDC dargestellt.

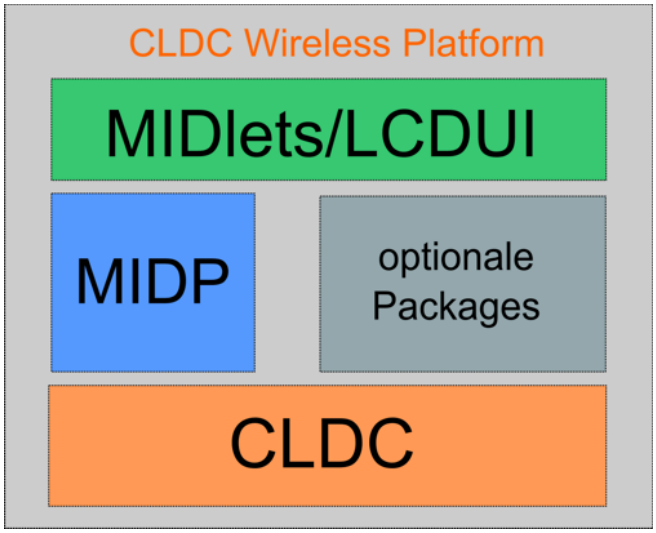

Abbildung 5.1: CLDC

5.1.1.2 Bluetooth

Die Technologie Bluetooth wurde bereits in Abschnitt 3.3 behandelt.
Die Funktionsweise, wie bei der Programmierung diese Technologie eingesetzt wird, wird im folgenden Abschnitt behandelt.

5.1.2 Ablauf

Diese Forschungsstellung basiert auf folgenden Informationen:

- Teilnehmer

  Teilnehmer dieser Darstellung sind zwei mobile Geräte, die miteinander interagieren. Diese Applikation dient nur zur Illustration, daher ist die Funktionalität nur für zwei Geräte implementiert.

  Es ist grundsätzlich nur eine Applikation zu installieren, welche beide Grundfunktionialitäten zur Verfügung stellt. Diese sind:

  - **Client bzw. Prover**

    Der Prover (laut Protokoll) möchte beweisen, dass dieser das gemeinsame Geheimnis kennt, ohne Informationen darüber öffentlich zu machen. Dieser folgt dem Protokoll und führt die dort erwähnten Schritte durch.

    Der Prover hat im Vergleich zum Verifier die ”schwerere” Aufgabe, sprich den höheren Rechenaufwand.

  - **Server bzw. Verifier**

    Der Verifier (laut Protokoll) lässt sich den Beweis zeigen und verifiziert, ob der Prover das Geheimnis kennt bzw. authentifiziert werden kann.

    Die Durchführung erfolgt wie im Protokoll festgelegt.

  Der Server (Verifier) dient dazu, die Nachrichten vom Prover (Client) zu überprüfen und zu verifizieren, ob dieser das Geheimnis (vlg. ZK) kennt und somit authentifiziert werden kann.

  Da, wie bereits erwähnt, dieses Beispiel rein zur Illustration von ZK dient, besteht die Hauptaufgabe des Servers darin, die Authentifizierungsversuche vom Prover anzunehmen und je nachdem diesen zu authentifizieren bzw. abzulehnen.

Der Client (Prover) will vom Server authentifiziert werden indem er das Geheimnis zu beweisen versucht. Hierbei wird die Protokollprozedur durchgeführt und dem Verifier die gewünschten Antworten übersendet.

- Ablauf

  Der Ablauf des Testlaufs geht wie folgt von statten (Hinweis: Die Dokumentation der verwendeten Objekte der Bluetooth-API sind in [Mot08a] und [Sun06] nachzulesen):

  1. [VERIFIER] Verifier-Thread starten

     Der Verifier-Thread wird über den Menüeintrag "Be the Verifier" ausgewählt.

     Im Sourcecode sieht dies wie folgt aus:

     Listing 5.1: Verifier-Thread starten

```
1                Thread t=new Thread()
2                {
3                    public void run()
4                    {
5                        while(!isClosed)
6                        {
7                            conn = null;
8
9                            try {
10                               //listens to connections, blocks as long no
11                               //connection is established
12                               conn = notifier.acceptAndOpen();
13                               txtMessages.setString("Connection established...");
14                               switchDisplay(txtMessages);
15                               verify(conn);
16                           }
17                           catch (Exception e)
18                           {
19                               //continue even if something went wrong
20                           }
21                       }
22                   }
23               };
24               //starting thread
25               t.start();
```

     Hierbei wird durch die Methode *acceptAndOpen()* vom StreamConnectionNotifier die Connection geöffnet und auf eine Verbindung gewartet.

     Nachdem eine Verbindung zustande gekommen ist, wird die Methode *verify* aufgerufen.

  2. [PROVER] Bluetooth-Geräte suchen

     Die erste Aufgabe des Provers besteht darin, dass dieser verfügbare Bluetooth-Geräte findet.

     Hierbei wird über die implementierte Bluetooth-API ein Device-Discovery gestartet. Dies erfolgt mithilfe von:

Listing 5.2: Device Discovery

```
1            try {
2                //start search
3                agent.startInquiry(DiscoveryAgent.GIAC,this);
4            }
5            catch (BluetoothStateException e)
6            {
7                txtMessages.setString("Starting Inquiry failed...");
8            }
```

Durch die Methode *startInquiry* vom DiscoveryAgent wird die Gerätesuche angestoßen.

Da die Klasse *ActionManager* das Interface DiscoveryListener implementiert (s. 8.1), müssen für das Device Discovery die Methoden *public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)* (welche aufgerufen wird, wenn ein Gerät erkannt wurde) und *public void inquiryCompleted(int discType)* (wenn die Suche abgeschlossen ist) implementiert werden.

Listing 5.3: deviceDiscovered

```
1        /*
2         * public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
3         * Method inherited from DiscoveryListener. Is called, if a device
4         * has been disovered. If so, the device is added to a Vector.
5         * Methode geerbt von DiscoveryListener. Wird aufgerufen, wenn ein Ger"at
6         * gefunden wurde und f"ugt dieses in eine Vector hinzu.
7         */
8
9        public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
10       {
11           devices.addElement(btDevice);
12           try
13           {
14               txtMessages.setString(btDevice.getFriendlyName(true));
15           }
16           catch(Exception e){}
17       }
```

Listing 5.4: inquiryComplete

```
1        /*public void inquiryCompleted(int discType)
2         * inherited from DiscoveryListener. Is called, if the device discovery
3         * is complete.
4         * vom DiscoveryListener geerbt. Wird aufgerufen, wenn die Ger"atesuche
5         * abgeschlossen ist
6         */
7
8        public void inquiryCompleted(int discType)
9        {
10           txtMessages.setString("Device Discovery completed...");
11           switchDisplay(txtMessages);
12       }
```

3. [PROVER] Gewünschtes Gerät auswählen

   Nachdem die Gerätesuche abgeschlossen ist, wählt der Prover im Menü den Ein-

trag "Authenticate.." aus. Dadurch wird eine Liste der gefundenen Einträge angezeigt und ein Gerät (welches den Verifier darstellt) kann ausgewählt werden.

4. [PROVER] Dienst suchen

   Nachdem dieses ausgewählt wurde, wird das zur Verfügung gestellte Service mithilfe der Methode *searchServices* des DiscoveryAgent gesucht.

Listing 5.5: searchServices

```
try
{
    //start search
    agent.searchServices(null,service, devToAuth, parent);
}
catch(Exception e)
{
    //
}
```

Auch hierbei müssen aufgrund des Interfaces DiscoveryListener Methoden implementiert werden.

Listing 5.6: Beispielcode

```
/*public void servicesDiscovered(int transID, ServiceRecord[] servRecord)
 * inherited from DiscoveryListener. Is called, if on the device, which
 * is searched on, services has been discovered. If a service is found,
 * the method prove with the url of the service is called (verifier
 * in the protocol)
 * von DiscoveryListener geerbt. Wird aufgerufen, wenn auf dem Ger"at Dienste
 * gefunden wurden. Falls ein Dienst gefunden wurde, wird mit dessen URL
 * die Funktion prove aufgerufen (der Verifier im Protokoll)
 */

public void servicesDiscovered(int transID, ServiceRecord[] servRecord)
{
    String serverurl="";
    txtMessages.setString("Services␣found...");
    for(int i=0;i<servRecord.length;i++)
    {
        record = servRecord[i];
        txtMessages.insert(serverurl=record.getConnectionURL(record.
            NOAUTHENTICATE_NOENCRYPT, false),txtMessages.size());
    }
    switchDisplay(txtMessages);
    prove(serverurl);
}
```

Listing 5.7: Beispielcode

```
/*
 * public void serviceSearchCompleted(int transID, int respCode)
 * inherited from DiscoveryListener. is called, if the service search is
 * complete.
 * von DiscoveryListener geerbt. Wird aufgerufen, wenn die Dienst-Suche
 * abgeschlossen ist.
 */

public void serviceSearchCompleted(int transID, int respCode)
```

```
32          {
33              //not implemented
34          }
```

5. [PROVER/VERIFIER] Ablauf nach Protokoll
   Nachdem die Verbindung zustande gekommen ist, erfolgt die Kommunikation nach Protokoll (s. 2.4.2) in den Methoden *prove* und *verify*.

5.1.2.1 Screens

Hier werden nun analog zur vorherigen Beschreibung des Ablaufs Screens der Applikation dargestellt:

- Start der Applikation - Abbildung 5.2
- Meldung nach Auswahl der Verifierrolle - Abbildung 5.3
- Gerätsuche - Meldung 1 - Abbildung 5.4
- Gerätsuche - Meldung 2 - Abbildung 5.5
- Authentifizierung - Liste der Geräte - Abbildung 5.6
- Meldung - Bluetooth-Sicherheit - Abbildung 5.7
- Meldung Ergebnis Prover/Verifier - s. 5.4
- weitere Screens - s. 8.2

5.2 Erwartetes Ergebnis

Das Ergebnis dieses Beispiels sollte ein erfolgreicher Authentifizierungsvorgang mithilfe des Fiat-Shamir Zero Knowledge-Protokolls sein.
Hierbei wird das Protokoll wie in 2.4.2 dargestellt umgesetzt und sollte ein mit einer Wahrscheinlichkeit von 100% positiv ausfallen.

Abbildung 5.2: Startbildschirm der Applikation

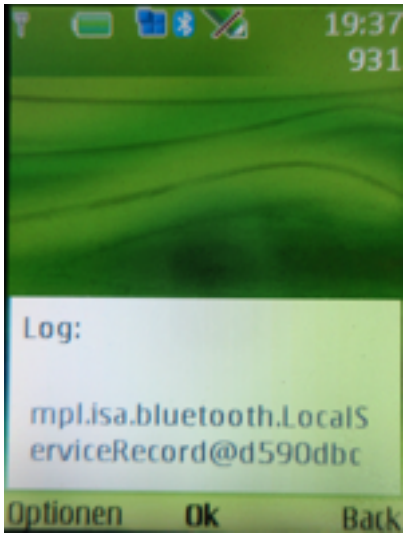

Abbildung 5.3: Auswahl der Verifierrolle

5.3 Durchführung

Die Durchführung erfolgt nach dem Muster von 5.1.2 und wird in folgender Reihenfolge ausgeführt:

1. Starten der Applikation auf beiden Geräten
2. Gerät 1 startet mithilfe der Menüwahl ”Be the Verifier..” den Verifier-Thread
3. Gerät 2 sucht nach Bluetooth-Geräten (Menüwahl ”Search Devices..”)
4. Gerät 2 startet über ”Authenticate..” den Authentifizierungsvorgang

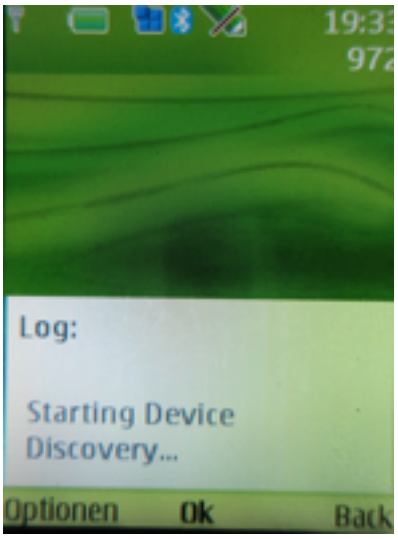

Abbildung 5.4: Gerätesuche - Meldung 1

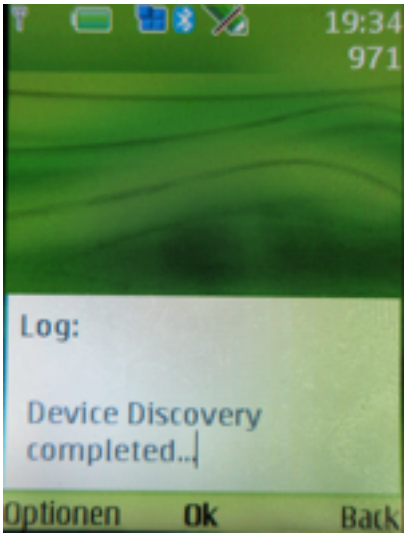

Abbildung 5.5: Gerätesuche - Meldung 2

5. Gerät 2 wählt aus der Liste der gefundenen Geräte das gewünschte aus

6. Der Verifiziervorgang wird gestartet (intern 10 Runden des Protokolls)

5.4 Ergebnis

Das Ergebnis des Testlaufs ist wie erwartet eine 100% ige Rate bei den Protokolldurchläufen.
Die Zeit für diese 10 Runden betragen < 1 s.
Abbildung 5.8 und 5.9 zeigen jeweils das Ergebnis von Prover und Verifier.

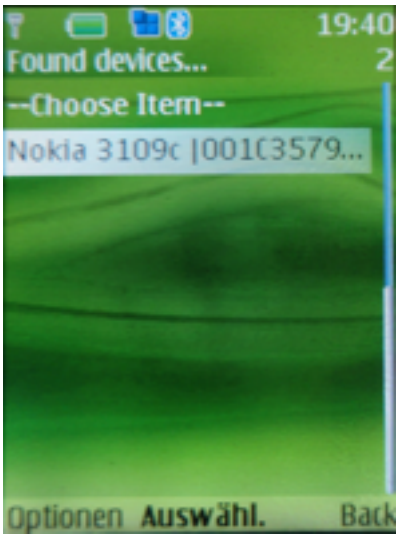

Abbildung 5.6: Authentifizierung - Liste der Geräte

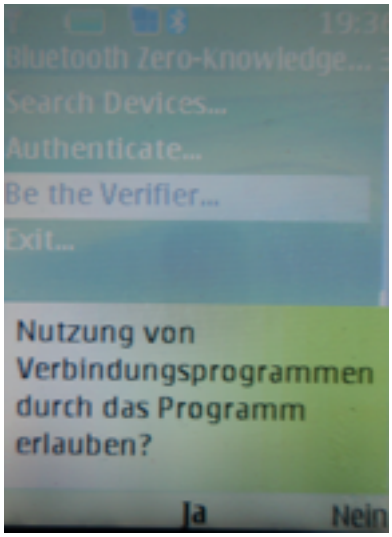

Abbildung 5.7: Meldung Bluetooth-Sicherheit

5.5 Fazit

Die Implementierung eines Zero Knowledge-Protokolls (hier Fiat-Shamir) geht relativ einfach vonstatten. Es gibt jedoch einige Regeln, die hier nicht miteingeflossen sind, wie z.B. die gewählten Primzahlen etc.

Mithilfe der Bluetooth-Technologie ist es jedoch relativ aufwendig bzw. dauert die Vorbereitungsphase durch die Geräte- bzw. Dienstsuche relativ lange, was bei anderen Technologien (wie z.B. NFC) nicht der Fall wäre, da hierbei die Gerätesuche wegfallen würde. Die Dienstsuche gestaltet sich relativ performant und ist im 100 ms-Bereich anzusiedeln.

Der Authentifizierungsvorgang (wie hier implementiert mit 10 Durchläufen) ist mit einer

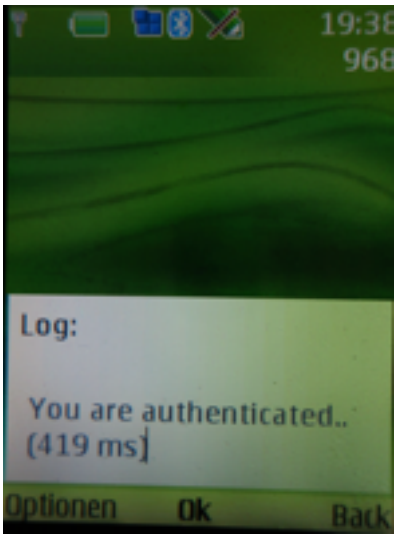

Abbildung 5.8: Ergebnis des Provers

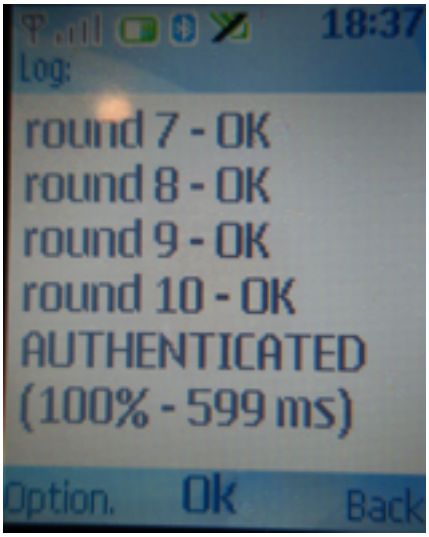

Abbildung 5.9: Ergebnis des Verifiers

Laufzeit von 1 s auch praktikabel. Hierbei muss jedoch auch beachtet werden, dass kleine Primzahlen verwendet wurden, die, je größer diese gewählt werden, mehr Aufwand bedeuten.

Abschließend ist zu sagen, dass, falls Zero Knowledge praktischen Einsatz findet, je nach Einsatzgebiet darauf zu achten ist, dass die Parameter (wie z.B. Primzahlen, Technologie, Protokoll) gut gewählt werden, sodass die Usability der Lösung nicht darunter leidet. Der Sicherheitsfaktor von Zero Knowledge wurde bereits geklärt, sodass dieser hier nicht besonders beachtet werden muss und durch das Protokoll abgedeckt ist.

Kapitel 6

Zukünftige Entwicklungen

Die Entwicklung von Zero Knowledge begann bereits vor 2 Jahrzehnten. Das Verfahren ist in theoretischer als auch in praktischer Weise bewiesen worden. Es wurden auch bereits Protokolle entworfen, die as-is umgesetzt und implementiert werden können. Zwar werden diese für die verschiedenen Einsatzzwecke modifiziert werden bzw. werden müssen, jedoch die Grundlage ist dieselbe.
Die Entwicklungen im Bereich Zero Knowledge könnten sein:

- Forschung im Bereich praktischer Einsatz
  Da der Einsatz von Zero Knowledge in der Praxis einerseits z.B. firmenintern erfolgt wird dieser Prozess nicht für die Öffentlichkeit freigegeben. Andererseits benötigt der Einsatz in der Praxis eine Vorbereitungs-, als auch Evaluierungsphase, in der die Parameter gewählt und untersucht werden.
  Speziell in diesem Bereich könnten universelle Parameter erforscht werden, die für etwaige reale Applikationen als Paradigma bzw. Template gelten können.

  Eine weitere Forschungsmöglichkeit kann speziell den Bereich des Einsatzes auf bzw. mit Chipkarten bieten, da hierbei die Möglichkeiten für den Praxiseinsatz reichlich gegeben sind.

- Marktreife
  Für den wirklichen Einsatz in der Praxis, sprich dass Zero Knowledge der breiten Masse einfach zugänglich ist, bedarf es weiterer Forschung und evtl. der Entwicklung eines eigenen Protokolls, welches dieses Kriterium erfüllt.

- Umstieg auf Zero Knowledge

  Falls ein Umstieg auf Zero Knowledge geplant sein sollte, muss in diese Richtung auch geforscht werden, ob und inwieweit die bisher eingesetzten Methoden abgelöst werden können oder ob die Ablösung wirklich die Vorteile von Zero Knowledge übernehmen kann.

Weiters wurde in dieser Arbeit der Themenbereich der parallelen Ausführung von Zero Knowledge nur kurz erwähnt und könnte zukünftig genauer betrachtet werden.

Kapitel 7

Conclusio

Heutzutage werden bei der Thematik der Authentifizierung verschiedene Methoden verwendet. Hierbei spielen unter anderem Passwortsysteme eine große Rolle, welches nicht als sicher gelten kann, da das Geheimnis (das Passwort) übermittelt und somit preisgegeben wird. Weitere Methoden betreffen die Authentifikation bzw. Identifikation mittels Biometrieverfahren etc., welche z.B. eine Eintrittsberechtigung erteilen.
Die Authentifizierung selbst ist ein wichtiger und absolut relevanter Teil der IT-Gesellschaft, da sich ein Angreifer als jemand anderer ausgeben könnte. Daher muss gewährleistet sein, dass wirklich die richtige Entität authentifiziert wird.
Challenge-Response-Verfahren sind eine Basisfunktionalität der Kryptographie. Hierbei wird einem Teilnehmer eine Aufgabe (Challenge) gestellt, die der andere durch Wissen lösen muss. Bei diesem Verfahren (zu dem auch grundsätzlich das Passwort-Verfahren zählt), wird, da ein Angreifer mithören könnte, dieses Verfahren durch Verschlüsselungsverfahren unterstützt.
Zero Knowledge ist eine Modifikation des Challenge-Response-Verfahrens, wobei die Zero Knowledge-Eigenschaft von Interesse ist. Diese besagt, dass während des Protokollablaufs keine Informationen über das Geheimnis preisgegeben werden und somit kein Angreifer für sich relevante Daten erhält.
Zero Knowledge basiert auf interaktiven Beweissystemen, wobei sich jemand von etwas überzeugen lassen kann. Diese Systeme werden bei Zero Knowledge erweitert, wobei neben den Bedingungen Completeness und Soundness die Zero Knowledge-Eigenschaft existiert.
Für die Anwendung bzw. den praktischen Einsatz von Zero Knowledge werden die Protokolle Fiat-Shamir, Feige-Fiat-Shamir, Guillou-Quisquater und Schnorr beschrieben und bewertet. Je nach Einsatzbereich sind diese Protokoll praktikabler als andere. Jedoch haben alle die

Eigenschaft, dass diese auf Chipkarten implementiert werden können, welches ein breites Einsatzspektrum aufwirft.

In mobilen Geräten werden heutzutage vielmals Kommunikationstechnologien wie Infrarot (IrDA), Bluetooth sowie NFC eingesetzt. Für den Einsatz von Zero Knowledge-Protokollen spielt hierbei die Applikationsentwicklungsmöglichkeit eine große Rolle. Aufgrund der Eigenschaften dieser Übertragungsmöglichkeiten kann die Praktikabilität beim Einsatz vom Zero Knowledge-Protokollen festgelegt werden. Hierbei gilt natürlich auch der Faktor, wie weit zu authentifizierende Geräte voneinander entfernt sind.

Die Einsatzmöglichkeiten von Zero Knowledge sind mitunter sehr weit gestreut, da die Authentifizierung ein weit verbreitetes Thema ist und in vielen Umgebungen eingesetzt werden kann. Anwendungsbeispiele sind mitunter der Einsatz bei digitalen Inhalten (DRM, Online-Inhalte etc.) als auch die klassische Authentifizierung in IT-Systemen (z.B. an Servern). Da Zero Knowledge eine Abwandlung des Challenge-Response-Verfahrens ist, kann auch dessen Einsatzbereich teilweise übernommen werden.
Die Implementierung eines Zero Knowledge-Protokolls auf Basis von Bluetooth-fähigen Geräten gestaltet sich mittels Java sehr einfach, da diese strikt nach Protokoll erfolgt. Durch die Java-Technologie ist auch der Zugriff auf die Bluetooth-Hardware problemlos möglich.
Beim konkreten praktischen Einsatz von Zero Knowledge muss jedoch je nach System eine gute Parameterwahl erfolgen, um die optimale Usability garantieren zu können. Weiters können diese Protokollen für die Verwendungszwecke angepasst werden.
Abschließend ist zu sagen, dass, falls in Zukunft konkrete Forschung im Bereich Zero Knowledge stattfindet um diese Technologie auch praktisch umzusetzen, diese sich durchaus etablieren kann und eine Alternative zu den bisher eingesetzten Authentifizierungs- und Identifizierungsmethoden darstellt.

Kapitel 8

Appendix

8.1 Sourcecode des Beispiels

8.1.1 BtZKApp.java

Listing 8.1: BtZKApp.java

```
1  import javax.microedition.midlet.*;
2  import javax.microedition.lcdui.*;
3
4  /**
5   * public class BtZKApp
6   * @author Markus Brandl, BSc.
7   * This class is a MIDlet which implements the functionality to demonstrate
8   * the use of a Zero Knowledge Protocol (the Fiat-Shamir Protocol in concrete).
9   * Hereby the user has the possibility to choose the role in the authentication
10  * process (prover or verifier) by selecting the appropriate menu item.
11  * Diese Klasse ist ein MIDlet, welches die Funktionalit"at zur Demonstration der
12  * Verwendung eines Zero Knowledge-Protokolls (das Fiat-Shamir-Protokoll)
13  * implementiert.
14  * Hierbei hat der Benutzer die M"oglichkeit die Rolle im Authentifizierungsvorgang
15  * zu w"ahlen (Beweiser oder Verifizierer), indem das passende Men"uelement ausgew"ahlt
16  * wird.
17  */
18 public class BtZKApp extends MIDlet implements CommandListener
19 {
20     private Command okCommand;
21     private Command exitCommand;
22     private List menu;
23     private Display disp;
24
25     /**
26      * public void startApp()
27      * called on MIDlet startup, creates the menu
28      * wird beim Starten des MIDlets aufgerufen, stellt das Men"u dar
29      */
30
31     public void startApp() {
32         String[] menuelements={"Search Devices...","Authenticate...","Be the Verifier...","Exit...
               "};
```

```
33          okCommand=new Command("Ok", Command.OK,0);
34          exitCommand=new Command("Exit", Command.EXIT,0);
35          menu=new List("Bluetooth Zero-Knowledge - Choose Action:",List.IMPLICIT,menuelements,null)
                ;
36          menu.addCommand(okCommand);
37          menu.addCommand(exitCommand);
38          menu.setCommandListener(this);
39          disp=Display.getDisplay(this);
40          disp.setCurrent(menu);
41      }
42
43      public void pauseApp() {
44          //not implemented
45      }
46
47      public void destroyApp(boolean unconditional) {
48          //not implemented
49      }
50
51      /**
52       * public void commandAction(Command c, Displayable d)
53       * listens for user input an reacts by calling a method in class ActionManager
54       * reagiert auf  Benutzereingabe und ruft jeweils eine Methode in der Klasse
55       * ActionManager auf
56       */
57
58      public void commandAction(Command c, Displayable d)
59      {
60          if(d==menu) //if selection is from menu
61          {
62              if(c==exitCommand)  //if exit
63              {
64                  disp.setCurrent(null);
65                  destroyApp(true);
66                  notifyDestroyed();
67              }
68
69              ActionManager.setDisplay(disp,menu);
70              switch (menu.getSelectedIndex())   //choose action based on item selection
71              {
72                 case 0: ActionManager.doIt(0);  //device discovery
73                     break;
74                 case 1: ActionManager.doIt(1);  //authenticate
75                     break;
76                 case 2: ActionManager.doIt(2);  //be the prover
77                     break;
78                 case 3: destroyApp(true);       //exit application
79                         notifyDestroyed();
80                     break;
81              }
82          }
83      }
84  }
```

8.1.2 ActionManager.java

Listing 8.2: ActionManager.java

```
1  import javax.microedition.lcdui.*;
2  import javax.microedition.io.*;
3  import javax.bluetooth.*;
```

```
4  import java.io.*;
5  import java.util.Vector;
6  import java.util.Random;
7
8  /**
9   * public class ActionManager
10  * @author Markus Brandl, BSc.
11  * This class provides the main functionality for the Zero Knowledge Protocol
12  * and manages the data transfer as well as the communication between two devices.
13  * Diese Klasse stellt die Funktionalit"at f"ur das Zero Knowledge-Protokoll zur
14  * Verf"ugung und steuert den Datentransfer und die Kommunikation zwischen zwei
15  * Ger"aten
16  */
17 public class ActionManager implements DiscoveryListener
18     {
19         private static TextBox txtMessages=new TextBox("Log:","",1000,TextField.UNEDITABLE);
20         private static Command okCommand;
21         private static Command backCommand;
22         private static Display disp;
23         private static LocalDevice localDevice;
24         private static DiscoveryAgent agent;
25         private static Displayable da;
26         private static Vector devices=new Vector();
27         public List lstDevices;
28         public RemoteDevice devToAuth;
29         private static final UUID SERVICE_UUID =
30                   new UUID("F0E0D0C0A0001",false);
31         private static final int roundNum=10;
32         public boolean isClosed=false;
33         private StreamConnection conn;
34         private StreamConnectionNotifier notifier;
35         private ServiceRecord record;
36         public ActionManager parent=this;
37         private static int n=21;
38
39  //MAIN FUNCTION
40         /*
41          * public static void doIt(int operation)
42          * calls the appropriate function based on the given parameter
43          * ruft die dazugeh"orige Funktion basierend auf dem Parameter auf
44          */
45
46         public static void doIt(int operation)
47         {
48             ActionManager am=new ActionManager();
49
50              switch(operation)
51              {
52                  case 0: am.SearchDevices();
53                      break;
54                  case 1: am.Authenticate();
55                      break;
56                  case 2: am.StartServer();
57              }
58         }
59
60  //DISPLAY FUNCTIONS
61
62         /*
63          * public static void setDisplay(Display d, Displayable da_old)
64          * sets the Displayable object onto the Display by the help of init()
65          * setzt das Displayable Objekt auf das Display, wobei die Funktion init()
66          * verwendet wird
67          */
68         public static void setDisplay(Display d,Displayable da_old)
```

```
69          {
70              disp=d;
71              da=da_old;
72              init();
73          }
74
75          /*
76           * public static void switchDisplay(Displayable d)
77           * sets the Displayable d (parameter) as the current object on the screen
78           * setzt das Displayable d (Parameter) als das aktuell sichtbare Objekt
79           */
80          public static void switchDisplay(Displayable d)
81          {
82              disp.setCurrent(d);
83          }
84
85          /*
86           * public static void init
87           * adds the CommandListener to the menu and the navigation items respectively
88           * setzt den CommandListener auf das Men"u bzw. die Navigationselemente
89           */
90
91          public static void init()
92          {
93              okCommand=new Command("Ok", Command.OK,0);
94              backCommand=new Command("Back", Command.BACK,0);
95              txtMessages.addCommand(okCommand);
96              txtMessages.addCommand(backCommand);
97              txtMessages.setCommandListener(new CommandListener(){
98                  public void commandAction(Command c, Displayable d)
99                  {
100                     if(c.getLabel().equals("Back"))
101                     {
102                         txtMessages.setString("Back");
103                         disp.setCurrent(da);
104                     }
105                     else if(c.getLabel().equals("Ok"))
106                     {
107                         txtMessages.setString("Ok");
108                         disp.setCurrent(da);
109                     }
110                 }
111             });
112         }
113
114 //SERVICE- AND DEVICE DISCOVERY
115
116         /*
117          * public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
118          * Method inherited from DiscoveryListener. Is called, if a device
119          * has been disovered. If so, the device is added to a Vector.
120          * Methode geerbt von DiscoveryListener. Wird aufgerufen, wenn ein Ger"at
121          * gefunden wurde und f"ugt dieses in eine Vector hinzu.
122          */
123
124         public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
125         {
126             devices.addElement(btDevice);
127             try
128             {
129                 txtMessages.setString(btDevice.getFriendlyName(true));
130             }
131             catch(Exception e){}
132         }
133
```

```
134         /*public void inquiryCompleted(int discType)
135          * inherited from DiscoveryListener. Is called, if the device discovery
136          * is complete.
137          * vom DiscoveryListener geerbt. Wird aufgerufen, wenn die Ger"atesuche
138          * abgeschlossen ist
139          */
140
141         public void inquiryCompleted(int discType)
142         {
143             txtMessages.setString("Device Discovery completed...");
144             switchDisplay(txtMessages);
145         }
146
147         /*public void servicesDiscovered(int transID, ServiceRecord[] servRecord)
148          * inherited from DiscoveryListener. Is called, if on the device, which
149          * is searched on, services has been discovered. If a service is found,
150          * the method prove with the url of the service is called (verifier
151          * in the protocol)
152          * von DiscoveryListener geerbt. Wird aufgerufen, wenn auf dem Ger"at Dienste
153          * gefunden wurden. Falls ein Dienst gefunden wurde, wird mit dessen URL
154          * die Funktion prove aufgerufen (der Verifier im Protokoll)
155          */
156
157         public void servicesDiscovered(int transID, ServiceRecord[] servRecord)
158         {
159             String serverurl="";
160             txtMessages.setString("Services found...");
161             for(int i=0;i<servRecord.length;i++)
162             {
163                 record = servRecord[i];
164                 txtMessages.insert(serverurl=record.getConnectionURL(record.
                        NOAUTHENTICATE_NOENCRYPT, false),txtMessages.size());
165             }
166             switchDisplay(txtMessages);
167             prove(serverurl);
168         }
169
170         /*
171          * public void serviceSearchCompleted(int transID, int respCode)
172          * inherited from DiscoveryListener. is called, if the service search is
173          * complete.
174          * von DiscoveryListener geerbt. Wird aufgerufen, wenn die Dienst-Suche
175          * abgeschlossen ist.
176          */
177
178         public void serviceSearchCompleted(int transID, int respCode)
179         {
180             //not implemented
181         }
182
183  //ZERO KNOWLEDGE PROTOCOL
184
185         /*
186          * public void StartServer()
187          * starts a server thread, which offers a connection based on a service url
188          * startet eine Server-Thread, welcher eine Verbindung basierend auf eine
189          * Service-URL zur Verf"ugung stellt
190          */
191
192         public void StartServer()
193         {
194             //building of the service url
195             StringBuffer url=new StringBuffer("btspp://localhost:");
196             url.append(SERVICE_UUID.toString());
197             url.append(";name=ZK Server;authorize=false;encrypt=false;master=false");
```

```
198
199             try
200             {
201                 localDevice = LocalDevice.getLocalDevice();
202                 localDevice.setDiscoverable(DiscoveryAgent.GIAC);
203                 notifier = (StreamConnectionNotifier) Connector.open(url.toString());
204
205                 //start of the "prover" mechanism
206                 record = localDevice.getRecord(notifier);
207                 txtMessages.setString("started...."+record.toString());
208                 switchDisplay(txtMessages);
209
210                 //creating thread
211                 Thread t=new Thread()
212                 {
213                     public void run()
214                     {
215                         while(!isClosed)
216                         {
217                             conn = null;
218
219                             try {
220                                 //listens to connections, blocks as long no
221                                 //connection is established
222                                 conn = notifier.acceptAndOpen();
223                                 isClosed=true;
224                                 txtMessages.setString("Connection established...");
225                                 switchDisplay(txtMessages);
226                                 verify(conn);
227                             }
228                             catch (Exception e)
229                             {
230                                 //continue even if something went wrong
231                             }
232                         }
233                     }
234                 };
235                 //starting thread
236                 t.start();
237             }
238             catch(Exception e)
239             {
240                 e.printStackTrace();
241             }
242         }
243
244         /*
245          * private void prove(String serverurl)
246          * does the full actions the prover (Zero Knowledge Protocol) has to do
247          * erledigt alle Aufgaben des Provers (Zero Knowledge-Protokoll)
248          */
249
250         private void prove(String serverurl)
251         {
252             DataOutputStream out=null;
253             DataInputStream in=null;
254             int p,q,s,v,x=1024,y=0,authcount=0;
255             boolean e=false, authenticated=false;
256             Random rand=null;
257             long timerstart=0, timerstop=0;
258
259             try
260             {
261                 txtMessages.setString("[VERIFIER] starting authentication ("+roundNum+"rounds)\n")
                        ;
```

```
262                 switchDisplay(txtMessages);
263                 //open connection
264                 StreamConnection connection = (StreamConnection) Connector.open(serverurl);
265                 //open data I/O streams
266                 out=connection.openDataOutputStream();
267                 in=connection.openDataInputStream();
268                 p=3;q=7;s=5;
269                 n=p*q;
270                 v=(s*s)%n;
271                 //measure the time
272                 timerstart=System.currentTimeMillis();
273                 //do the protocol action roundNum times
274                 for(int i=0;i<roundNum;i++)
275                 {
276                     //create random number
277                     rand=new Random(System.currentTimeMillis());
278                     //random boolean
279                     e=((rand.nextInt()%2)!=0);
280                     //read x from prover
281                     x=in.readInt();
282                     //send bit e
283                     out.writeBoolean(e);
284                     out.flush();
285                     //read y from prover
286                     y=in.readInt();
287                     //verify messages sent
288                     if(e==false)
289                     {
290                         if(((y*y)%n) == (x%n))
291                         {
292                             authenticated=true;
293                             authcount++;
294                             txtMessages.setString(txtMessages.getString()+"round␣"+(i+1)+"␣-␣OK\n"
                                    );
295                         }
296                         else
297                             txtMessages.setString(txtMessages.getString()+"round␣"+(i+1)+"␣-␣neg.\
                                    n");
298                     }
299                     else
300                     {
301                         if(((y*y)%n) == ((x*v)%n))
302                         {
303                             authenticated=true;
304                             authcount++;
305                             txtMessages.setString(txtMessages.getString()+"round␣"+(i+1)+"␣-␣OK\n"
                                    );
306                         }
307                         else
308                             txtMessages.setString(txtMessages.getString()+"round␣"+(i+1)+"␣-␣neg.\
                                    n");
309                     }
310                 }
311                 if(authcount>=(int)(roundNum*0.95))
312                 {
313                     timerstop = System.currentTimeMillis();
314                     long timerdiff=timerstop-timerstart;
315
316                     txtMessages.setString(txtMessages.getString()+"AUTHENTICATED␣("+(int)(
                            authcount/roundNum*100)+"%␣-␣"+timerdiff+"␣ms)");
317                     out.writeBoolean(true);
318                 }
319                 else
320                 {
321                     out.writeBoolean(false);
```

```
322                 }
323                 out.flush();
324 
325                 try
326                 {
327                     //close connection
328                     connection.close();
329                 }
330                 catch (Exception exc2) {
331                     txtMessages.setString("Connection could not been closed.."+exc2.getMessage());
332                 }
333             }
334             catch(Exception exc)
335             {
336                 txtMessages.setString("[VERIFIER] Error: "+exc.getMessage());
337                 switchDisplay(txtMessages);
338             }
339 
340         }
341 
342         /*
343          * private void verify(StreamConnection conn)
344          * function of the prover (Zero Knowledge)
345          * Funktion, welche den Prover darstellt
346          */
347 
348         private void verify(StreamConnection conn) {
349             DataOutputStream out=null;
350             DataInputStream in=null;
351             int r,s=5,x,y;
352             boolean e=false;
353             Random rand=null;
354             long timerstart=0, timerstop=0;
355 
356             try
357             {
358                 txtMessages.setString("[PROVER] starting authentication("+roundNum+"rounds)\n");
359                 switchDisplay(txtMessages);
360                 out=conn.openDataOutputStream();
361                 in=conn.openDataInputStream();
362                 //for alternative selection of r
363                 Vector znstar = CalcQuadRest(n);
364 
365                 timerstart=System.currentTimeMillis();
366 
367                 for(int i=0;i<roundNum;i++)
368                 {
369                     //get random number r
370                     rand=new Random(System.currentTimeMillis());
371                     r=rand.nextInt()%(n-1);
372                     if(r<0)r*=-1;
373                     //alternative selection of r
374                     /*Integer tmp=(Integer)znstar.elementAt(r);
375                     r=tmp.intValue();
376                     v=(s*s)%n;*/
377                     //calculate x
378                     x=(r*r)%n;
379                     //send x to verifier
380                     out.writeInt(x);
381                     out.flush();
382                     //read bit from verifier
383                     e=in.readBoolean();
384                     //send values based on bit
385                     if(e==false)
386                     {
```

```
387                         y=r%n;
388                         out.writeInt(y);
389                         out.flush();
390                     }
391                     else
392                     {
393                         y=((r*s)%n);
394                         out.writeInt(y);
395                         out.flush();
396                     }
397                 }
398                 //read authenticated bit
399 
400                 boolean authed=in.readBoolean();
401                 txtMessages.setString("seas␣"+authed);
402                 if(authed==true)
403                 {
404                     txtMessages.setString("You␣are␣authenticated..");
405                 }
406             }
407             catch(Exception exc)
408             {
409                 txtMessages.setString("[PROVER]␣Error:␣"+exc.getMessage());
410                 switchDisplay(txtMessages);
411             }
412             timerstop = System.currentTimeMillis();
413             long timerdiff=timerstop-timerstart;
414             txtMessages.setString(txtMessages.getString()+"␣("+timerdiff+"␣ms)");
415         try {
416             conn.close();
417         }
418         catch (Exception exc2) {
419            txtMessages.setString("Connection␣could␣not␣been␣closed.."+exc2.getMessage());
420         }
421     }
422 
423         /*
424          * public boolean SearchDevices()
425          * starts the device search
426          * statet die Ger"atesuche
427          */
428 
429         public boolean SearchDevices()
430         {
431             txtMessages.setString("Starting␣Device␣Discovery...");
432             switchDisplay(txtMessages);
433 
434             try {
435                 localDevice = LocalDevice.getLocalDevice();
436                 agent = localDevice.getDiscoveryAgent();
437             }
438             catch (Exception e)
439             {
440                 txtMessages.setString("Bluetooth␣Initialization␣failed...");
441                 return false;
442             }
443 
444             try {
445                 //start search
446                 agent.startInquiry(DiscoveryAgent.GIAC,this);
447             }
448             catch (BluetoothStateException e)
449             {
450                 txtMessages.setString("Starting␣Inquiry␣failed...");
451             }
```

```
452             return false;
453         }
454
455         /*
456          * public boolean Authenticate
457          * starts the authentication procedure. prints a list of found devices
458          * and starts based on selection the service inquiry.
459          * startet die Authentifizierungsprozedur. gibt die gefundenen Ger"ate in
460          * einer Liste aus und startet die Dienstsuche basierend auf der Wahl
461          * des Ger"ats
462          */
463
464         public boolean Authenticate()
465         {
466             lstDevices=new List("Found devices...",List.IMPLICIT,new String[]{"--Choose Item--"},
                     null);
467
468             for(int i=0;i<devices.size();i++)
469             {
470                 RemoteDevice remdev=(RemoteDevice)devices.elementAt(i);
471                 try
472                 {   //add found devices to List (display)
473                     lstDevices.append(remdev.getFriendlyName(false)+" |"+remdev.
                             getBluetoothAddress(), null);
474                 }
475                 catch(Exception ioe)
476                 {
477                  txtMessages.setString("Error adding Device to List...");
478                 }
479             }
480             //if no devices has been found, list would be empty, so no further actions
481             if(devices.isEmpty()==true)
482             {
483                 txtMessages.setString("No Devices found...");
484                 switchDisplay(txtMessages);
485             }
486             else
487             {
488                 Command okLstCommand=new Command("Ok", Command.OK,0);
489                 Command backLstCommand=new Command("Back", Command.BACK,0);
490
491                 //add Listener to List items
492                 lstDevices.addCommand(okLstCommand);
493                 lstDevices.addCommand(backLstCommand);
494                 lstDevices.setCommandListener(new CommandListener(){
495                 public void commandAction(Command c, Displayable d)
496                 {
497                     if(c.getLabel().equals("Back"))
498                     {
499                         txtMessages.setString("Back");
500                         disp.setCurrent(da);
501                     }
502                     else if(c.getLabel().equals("Ok"))
503                     {
504                         txtMessages.setString("Ok");
505                     }
506                     String btString=lstDevices.getString(lstDevices.getSelectedIndex());
507                     //get Bluetooth adress
508                     int index=btString.indexOf(" |");
509                     String btAdr=btString.substring(index+2);
510                     for(int i=0;i<devices.size();i++)
511                     {
512                         RemoteDevice cur=(RemoteDevice)devices.elementAt(i);
513                         if(cur.getBluetoothAddress().equals(btAdr))
514                         {
```

```
515                             lstDevices.append("curdev set:"+btAdr, null);
516                             devToAuth = cur;
517                         }
518                     }
519                     if(devToAuth!=null)
520                     {
521                         //search for service with fix UUID
522                         UUID temp=new UUID("F0E0D0C0A0001",false);
523                         UUID[] service=new UUID[1];
524                         service[0]=temp;
525                         txtMessages.setString("Starting Service Discovery...");
526                         switchDisplay(txtMessages);
527                         try
528                         {
529                             //start search
530                             agent.searchServices(null,service, devToAuth, parent);
531                         }
532                         catch(Exception e)
533                         {
534                             e.printStackTrace();
535                         }
536                     }
537                 }
538             });
539                 switchDisplay(lstDevices);
540             }
541 
542             return false;
543         }
544 
545         /*
546          * private Vector CalcQuadRest(int n)
547          * alternative calculation of r (see Zero Knowledge protocol)
548          * calculates the elements of Zn*
549          * alternative Berechnung von r (vgl. Zero Knowledge-Protokoll)
550          * berechnet die quadratischen Reste von n (Zn*)
551          */
552         private Vector CalcQuadRest(int n)
553         {
554             Vector v=new Vector();
555             int tmp;
556 
557             for(int i=0;i<n;i++)
558             {
559                 tmp=(i*i)%n;
560                 v.addElement(Integer.valueOf(tmp+""));
561             }
562             return v;
563         }
564     }
```

8.2 Alle Screens des Ablaufs der Applikation

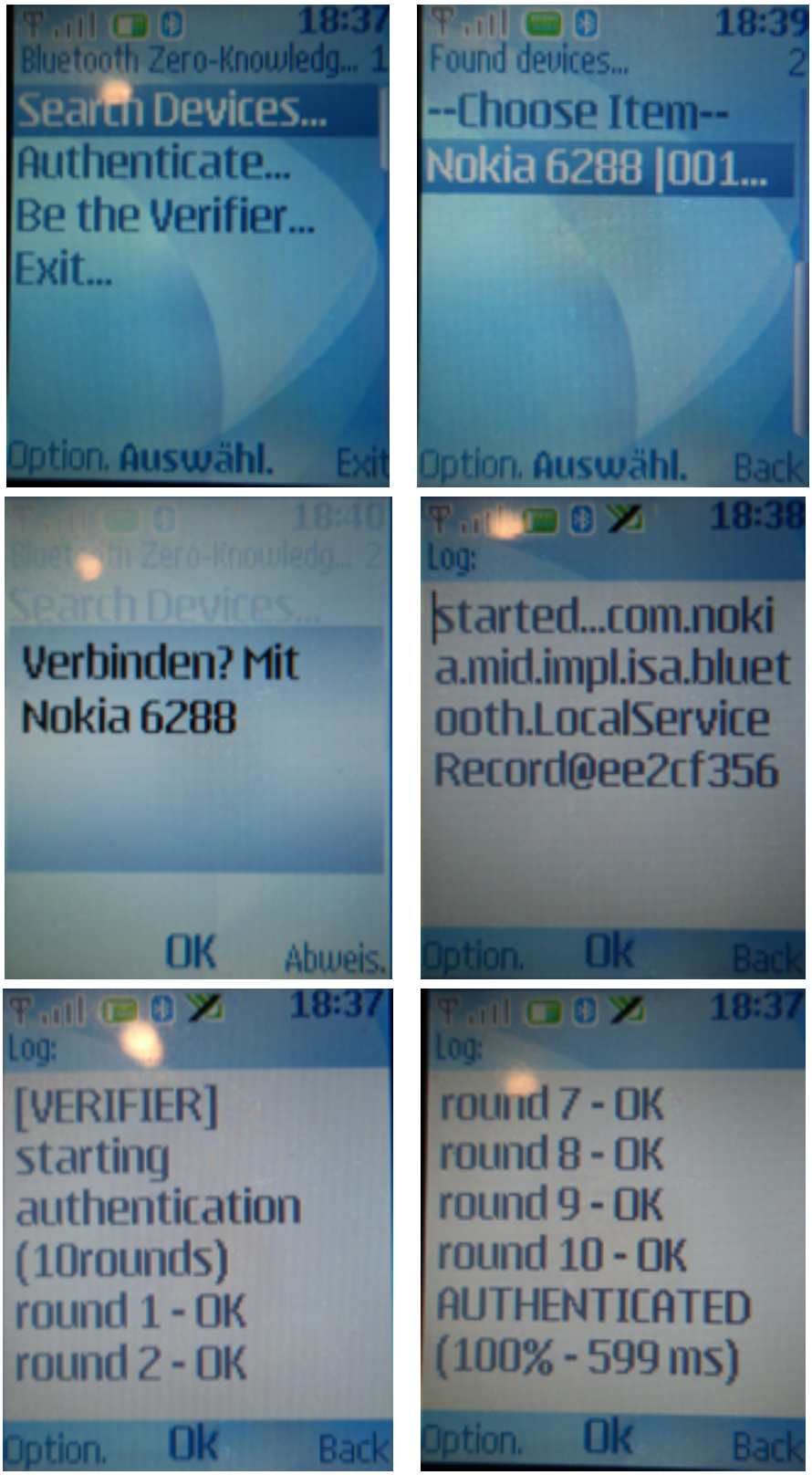

Abbildung 8.1: Screens der Applikation

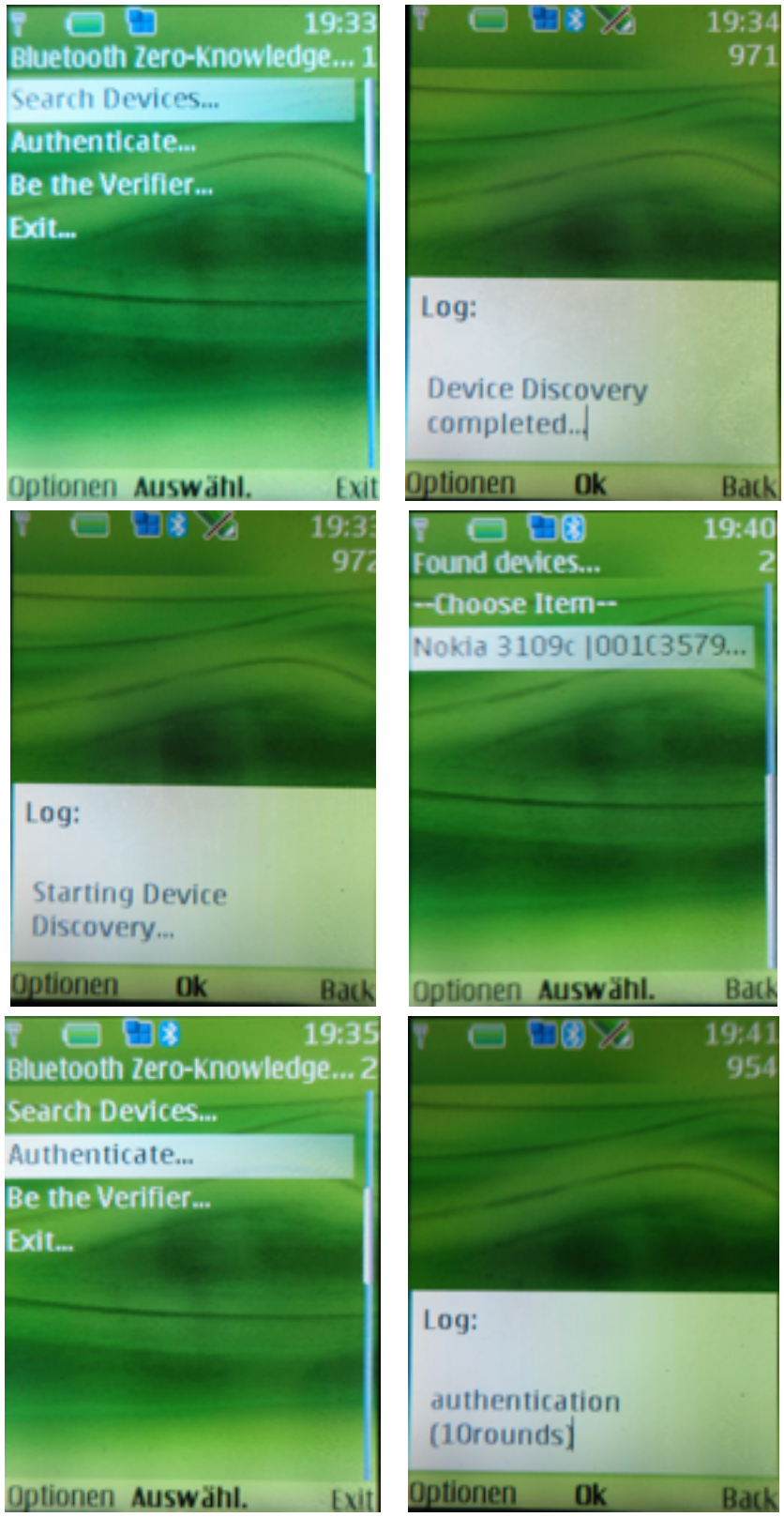

Abbildung 8.2: Screens der Applikation - Teil 2

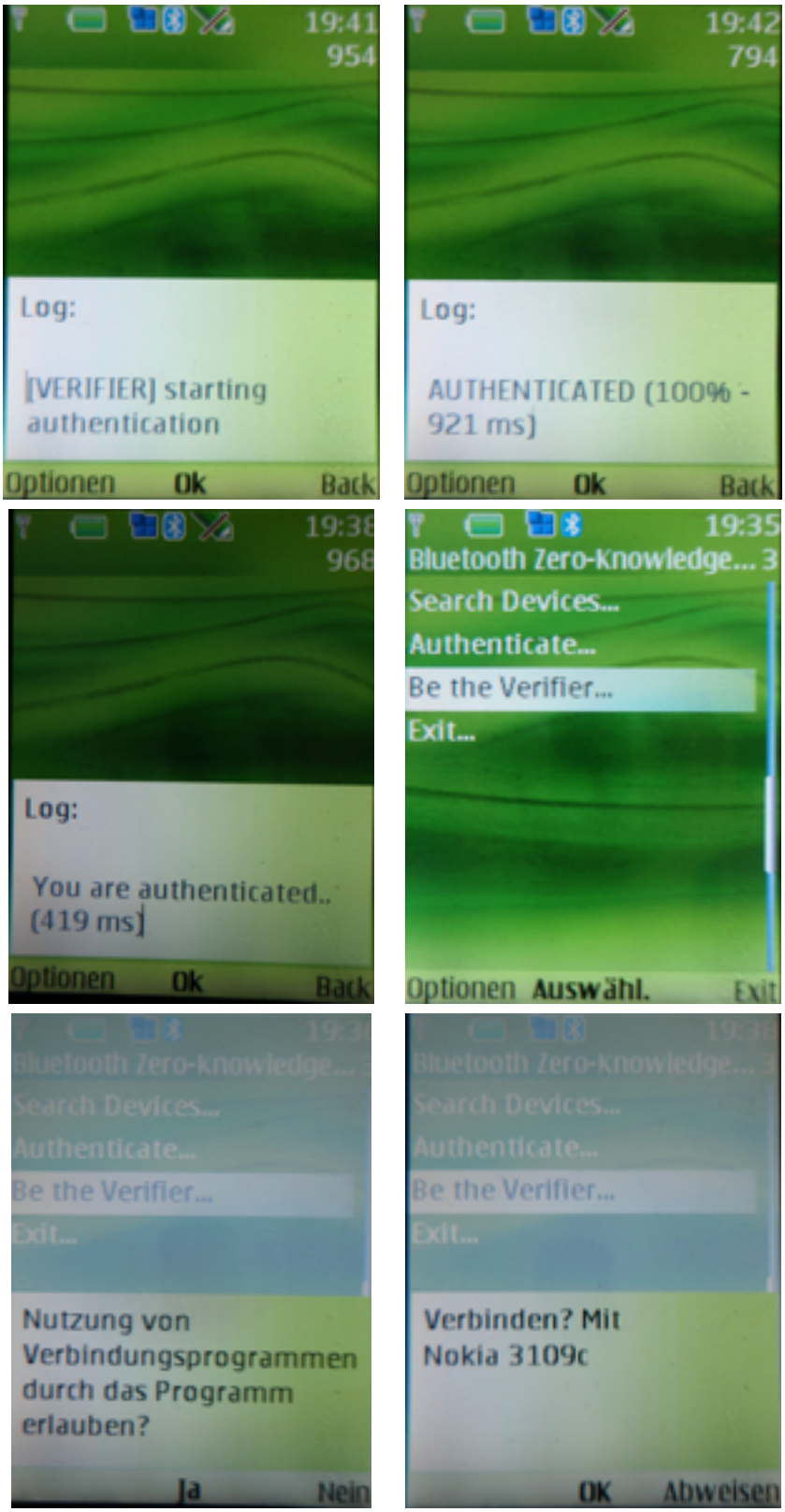

Abbildung 8.3: Screens der Applikation - Teil 3

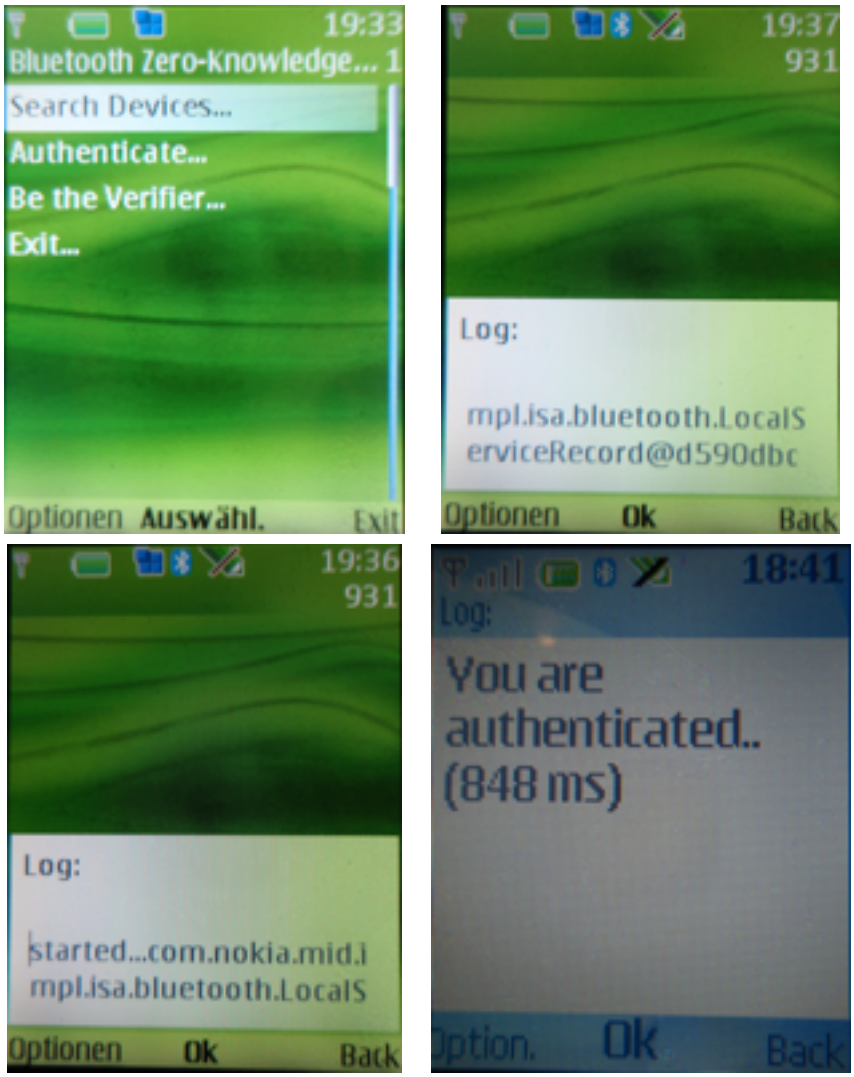

Abbildung 8.4: Screens der Applikation - Teil 4

Literaturverzeichnis

[ABI08] ABI Research, 2008, http://www.abiresearch.com/products/market_research/Near-Field_Communications_(NFC).

[Alb06] Albrecht Beutelspacher, Jörg Schwenk, Klaus-Dieter Wolfenstetter, *Moderne Verfahren der Kryptographie*, 6. auflage ed., Vieweg, Wiesbaden, 2006.

[Alf96] Alfred Menezes, Paul van Oorschot, Scott Vanstone, *Handbook of Applied Crytography*, 1st ed., CRC Press, 1996.

[Aus09] Mobilkom Austria, 2009, http://www.a1.net/privat/nfchandyshopping.

[Bar01] Boaz Barak, *How to go beyond the black-box simulation barrier*, In 42nd FOCS, 2001, pp. 106–115.

[blu07] bluetooth.com, *http://www.bluetooth.com/nr/rdonlyres/f8e8276a-3898-4ec6-b7da-e5535258b056/6545/core_v21_edr.zip*, online, Jul 2007.

[blu09a] bluetooth.com, 2009, http://bluetooth.com/Bluetooth/SIG/History_of_the_SIG.htm.

[blu09b] ———, *http://german.bluetooth.com/bluetooth/technology/works/compare/*, online, Jän 2009.

[blu09c] ———, *http://www.bluetooth.com/bluetooth/technology/building/specifications/*, online, Jän 2009.

[Bru03] Bruce Hopkins, Ranjith Antony , *Bluetooth for java*, Apress, 2003.

[ECM08] ECMA, *Wireless Proximity systems - Near Field Communication*, 2003-2008, http://www.ecma-international.org/publications/standards/Stnindex.htm.

[Ern06] Ernst Haselsteiner, Klemens Breitfuss, Jul 2006, http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002% 20-% 20Security% 20in% 20NFC.pdf.

[ets03] etsi.org, *http://etsi.org/website/standards/standardsdownload.aspx*, März 2003, ETSI TS 102 190.

[fre09] freeradius.org, *http://www.freeradius.org*, online, Jän 2009.

[Gol85] Rackoff Goldwasser, Micali, *The knowledge complexity of interactive proof-systems*, STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing (New York, NY, USA), ACM, 1985, pp. 291–304.

[IEE] IEEE, http://standards.ieee.org/getieee802/802.11.html.

[Int04] International Organization for Standardization, *Near Field Communication - Interface and Protocol (NFCIP-1)*, 2004, http://www.iso.org/iso/iso_catalogue.htm.

[Int08] ———, *Identification Cards, Contactless integrated curcuit cards, proximity cards*, 2008, http://www.iso.org/iso/iso_catalogue.htm.

[ird09] irda.org, 2009, http://irda.org/displaycommon.cfm?an=1.

[Kon07] Alan G. Konheim, *Computer Security and Cryptography*, John Wiley and Sons, Inc., Hoboken, New Jersey, 2007.

[Mot08a] Motorola, 2008, http://jcp.org/aboutJava/communityprocess/mrel/jsr082/index2.html.

[Mot08b] ———, *Java apis for bluetooth wireless technology (jsr 82)*, Juli 2008, http://jcp.org/en/jsr/detail?id=82.

[Mul01] Nathan J. Muller, *Bluetooth*, 1. auflage ed., MITP, Bonn, 2001.

[nfc] nfc-forum.org, *About NFC*, http://www.nfc-forum.org/aboutnfc.

[nfc09] ———, *http://www.nfc-forum.org*, online, Jän 2009.

[Nie03] Niels Ferguson, Bruce Schneier, *Practical cryptography*, Wiley Publishing, Inc., Indianapolis, 2003.

[Nok06] Nokia, *Jsr 257: Contactless communication api*, Okt 2006, http://jcp.org/en/jsr/detail?id=257.

[Ode91] Oded Goldreich, Silvio Micali, Avi Widgerson, *Proofs that Yield Nothing But Their Validity of All Languages in NP Have Zero-Knowledge Proof Systems*, vol. 38, Association for Computing Machinery, 1991.

[Ode01] Oded Goldreich, *Foundations of Cryptographoy - Basic Tools*, vol. 1, Cambridge University Press, 2001.

[Ode04] ———, *Zero-Knowledge twenty years after its invention*, 2004, http://www.wisdom.weizmann.ac.il/ oded/PS/zk-tut02v4.ps.

[Ode05] ———, *Foundation of Cryptography - A Primer*, now Publishers Inc., 2005.

[oS09] University of Stanford, 2009, http://qwiki.stanford.edu/wiki/Complexity_Zoo:B#bpp.

[Pac98] Hewlett Packard, Feb 1998, http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/hpjindex.htm

[Pac00] ———, Okt 2000, http://www.hpl.hp.com/personal/Jean_Tourrilhes/IrDA/Java.html.

[Ros06] Alon Rosen, *Concurrent zero-knowledge*, Springer, Berlin, Heidelberg, New York, 2006.

[Sch96] Bruce Schneier, *Applied cryptography*, second edition ed., John Wiley & Sons, Inc., New York, 1996.

[Sti95] Douglas Stinson, *Cryptography: Theory and Practice*, CRC Press, 1995.

[Sun06] Sun Microsystems, Motorola, 2006, http://java.sun.com/javame/reference/apis/jsr118/.

[Sun09] Sun Microsystems, 2009, http://java.sun.com/javame/technology/index.jsp.

[Tao06] Taoufik Ghanname, *How NFC can to speed Bluetooth transactions - today*, Feb 2006, http://www.wirelessnetdesignline.com/howto/180201430.

Abbildungsverzeichnis

2.1 ZK Höhle 8

2.2 Zwei isomorphe Graphen 11

2.3 Grand Chessmaster Problem 29

2.4 Mafia-Betrug 30

2.5 Terrorist-Betrug 31

3.1 IrDA-Stack 35

3.2 Bluetooth-Netzwerkarten 37

3.3 Bluetooth-Stack 38

3.4 GAP - Generic Access Profile 40

3.5 NFC Modi und dazugehörige Übertragungsraten 45

3.6 NFC Protokollablauf 47

4.1 Zero Knowledge als Authentifikationsschicht 55

5.1 CLDC 62

5.2 Startbildschirm der Applikation 68

5.3 Auswahl der Verifierrolle 68

5.4 Gerätesuche - Meldung 1 69

5.5 Gerätesuche - Meldung 2 69

5.6 Authentifizierung - Liste der Geräte 70

5.7 Meldung Bluetooth-Sicherheit 70

5.8 Ergebnis des Provers 71

5.9 Ergebnis des Verifiers 71

8.1 Screens der Applikation 87

8.2 Screens der Applikation - Teil 2 88

8.3 Screens der Applikation - Teil 3 89

8.4 Screens der Applikation - Teil 4 90

Tabellenverzeichnis

3.1 Bluetooth-Klassen 37

3.2 Bluetooth-Eigenschaften 43

3.3 Vergleich der Technologien (in Anlehnung an [Tao06]) 49

Listings

5.1 Verifier-Thread starten 64

5.2 Device Discovery 65

5.3 deviceDiscovered 65

5.4 inquiryComplete 65

5.5 searchServices 66

5.6 Beispielcode 66

8.1 BtZKApp.java 76

8.2 ActionManager.java 77