**Dissertation**

# PbAE

**Picture based Authentication and Encryption**

**Bildbasierte Authentifizierung und Verschlüsselung**

Zur Erlangung des akademischen Grades Doktor rer. nat.

Vorgelegt dem Fachbereich Elektrotechnik/Informatik
der Universität Kassel

Von **Dipl.-Ing. Nabil Benamar**
geb. am 30.07.1972

Ausgabe und Betreuung durch
**Herrn Prof. Dr. Lutz Wegner**

Eingereicht: April 2008

Mündliche Prüfung: 21 Mai 2008

# Erklärung

*Hiermit versichere ich, dass ich die vorliegende Dissertation selbständig und ohne unerlaubte Hilfe angefertigt und andere als die in der Dissertation angegebenen Hilfsmittel nicht benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht. Kein Teil dieser Arbeit ist in einem anderen Promotions- oder Habilitationsverfahren verwendet worden.*

*Kassel, den 10. April 2008*

*Nabil Benamar*

# Vorwort

Der Einsatz der Informationstechnik in Wirtschaft und Verwaltung nimmt ständig zu, e-business und e-government sind inzwischen verbreitete Schlagworte. Hinter den Konzepten einer ins Internet verlagerten und unter Umständen sogar mobil stattfindenden Abwicklung von Geschäftsfällen und Verwaltungsakten stecken aber viele reale Herausforderungen und zum Teil ungelöste praktische Probleme. Insbesondere besteht ein Bedarf an innovativen Lösungen für die Aspekte Komfort und Sicherheit, nachdem die grundsätzliche Machbarkeit inzwischen nicht mehr bezweifelt wird.

Zentraler Punkt ist dabei das Identitätsmanagement [IUS03][MJ01][Jen03], das wiederum auf Authentifizierungs- und Verschlüsselungsverfahren aufsetzt. Neben den komplexen mathematischen Algorithmen, die dafür zum Einsatz kommen und weitgehend abgesichert sind, spielt die praktische Umsetzung in praktikable Verfahren eine große Rolle. In dem Maße, wie elektronische Transaktionen Alltag werden, nimmt leider auch die kriminelle Energie zu, über Schwachstellen der Umsetzung oder unkluges Verhalten der Anwender in die Systeme einzubrechen, Daten zu verfälschen oder abzuschöpfen, bzw. sonst wirtschaftlichen Schaden anzurichten.

In dieser Arbeit wird ein neu entworfenes Verfahren vorgestellt, das eine verläßliche Authentifizierung erleichtert und damit auch die Speicherung privater Daten sicherer macht. Dazu werden mehrere bekannte Verfahren gebündelt und neu kombiniert, wodurch Stärken eines Ansatzes Schwächen anderer Verfahren aufheben können.

Als Einleitung werden einige aus dem Alltag bekannte Authentifizierungsmethoden, wie Passwort, PIN, TAN, iTAN, Ausweise usw. erläutert. Ausserdem wird auf Verschlüsselungsalgorithmen eingegangen, die für einen sicheren Schutz der Daten zum Einsatz kommen. Ein kurzer Überblick über die meist eingesetzten Angriffsmethoden soll die Notwendigkeit neuer Sicherheitssysteme zeigen. Die in der Einleitung aufgeführten Verfahren werden in der Fachliteratur ausführlich behandelt, deswegen

wird hier auf eine genauere Erklärung verzichtet. Sie werden dennoch kurz beschrieben, um einige für die Arbeit relevanten Begriffe zu erläutern.

# Danksagung

Meinem wissenschaftlichen Betreuer Herrn Prof. Dr. Lutz Wegner möchte ich für die freundliche Überlassung des Themas und für seinen vielseitigen fachlichen Rat danken. Ich verdanke ihm auch für seine sachkundige und richtungsweisende Begleitetung in jeder Phase der Arbeit.

Ich danke auch den wissenschaftlichen Mitarbeitern der Fachgruppe Praktische Informatik Herrn Dipl.-Math Kai Schweinsberg und Herrn Dipl.-Inform. Sebastian Pape für den innovativen Gedankenaustausch.

Dankbar anerkennen möchte ich die technische Unterstützung von Herrn Dipl.-Ing. Michael Möller und Frau Irmgard Zarges.

Diese Arbeit widme ich meiner Familie.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Quellcodeverzeichnis

# Teil I

# Einführung und Problemstellung

# Kapitel 1

# Grundlagen der Authentifizierung

Wem als Kind die orientalische Geschichte „Ali Baba und die 40 Räuber“ aus der Sammlung „Tausendundeine Nacht“ vorgelesen wurde, der kam unbewußt mit dem Thema Authentifizierung in Berührung. Die Geschichte erzählt, wie Ali Baba mit Hilfe der klugen Sklavin Mardschana eine vierzigköpfige Räuberbande nach und nach bezwingt und in den Besitz ihres in einer Felsenhöhle versteckten Schatzes gelangt. Mardschana wird zum Dank freigelassen und mit dem Neffen Ali Babas verheiratet. In den Sprachgebrauch eingegangen ist dabei das Losungswort **„Sesam-öffne-Dich!“**, mit dem das Felsentor der Schatzkammer zu öffnen ist (zitiert aus [Hof]).

Auch in allen anderen Kulturkreisen gibt es weit in die Vergangenheit zurückreichende Erzählungen von geheimen oder verbotenen Pforten (etwa den Zugang zur Unterwelt, den der Höllenhund Kerberos in der griechischen Mythologie bewacht) und von versteckten Schätzen. Auf der Gegenseite finden sich dann magische Worte, geschickte Tarnungen und Verkleidungen, nachgemachte Schlüssel oder der Fund geheimer Karten, mit denen sich der Schutz aufheben läßt oder man Zugang zum Verborgenen findet.

Da es in der vorliegenden Arbeit um Verfahren zur leichteren Authentifizierung und Verschlüsselung geht, seien zunächst die Begriffe etwas näher eingekreist. Nach gängiger Definition (vgl. etwa [HL]) versteht man unter **Identifizierung**, bzw. **Identifikation** den Prozess, in dem die Identität einer Person bestimmt wird (in der Daten oder Aktivitäten mit einer bestimmten Identität verknüpft/assoziiert werden). Die Identifikation wird durchgeführt durch Erlangung eines Identifikators.

Wenn man sicher sein will, dann muss die Behauptung („assertion“) „Diese Person $P$ hat Identität $I$“ authentifiziert werden. Als **Authentifizierung** bezeichnet man den Prozess, in dem Sicherheit über eine (Identitäts-)Behauptung gewonnen wird. Der Gegencheck erfolgt gegen einen oder mehrere Authentikatoren. Als Authentikatoren (mehr dazu weiter unten) können dienen

**Wissen:** was eine Person weiß (Passwort, Geburtsnamen der Mutter, PIN)

**Besitz:** was eine Person hat (Ausweis, Schlüssel, Chipkarte)

**Biometrische Eigenschaften:** was eine Person ist (Fingerabdruck, Spracheigenheit, Iris)

Koch [HL] unterscheidet genauer zwischen Identitäten und Entitäten in dem Sinn, dass etwa eine Person je nach Rolle unterschiedliche Identitäten (und Attributen, die zu diesen gehören) annehmen kann, aber nur eine Entität darstellt. Entsprechend liefert nur die Biometrie dann die Entität, statt von einem Identifikator spricht er dann von Entifikatoren. Zu letzteren zählt er auch, was eine Person (im Unterbewußsein) tun kann, etwa eine Unterschrift leisten oder ein Wort in einem speziellen Rythmus eintippen. Diese etwas feinere Unterscheidung nehmen wir hier nicht vor.

## 1.1 Wissen

Das „Wissen“ ist eine der ältesten Methoden der Authentisierung. Jemandem wurde ein Geheimnis verraten, wie ein Wort, einen Satz oder einen Klopfrhythmus. Die sich zu legitimierende Person wird dann nach dem Geheimnis gefragt und wenn sie es verraten kann, geht man davon aus, dass sie berechtigt ist.

Heutzutage kommt dieses Verfahren immer noch zum Einsatz. Die bekanntesten Einsatzbereiche sind Computer, an denen man sich anmelden kann, wenn man das Passwort kennt, oder die Geldkarten, die eine Geheimnummer, auch PIN genannt, benötigen.

In manchen Bereichen werden persönliche und sehr private Daten als Identitätsnachweis benutzt. Bekannt sind der Mädchenname der Mutter, der Nachname der ersten Schulfreundin, die Marke des ersten eigenen Autos, der Spitzname eines Haustiers. Generell sind auch anwendergenerierte, eher ungewöhnliche Frage-Antwort-Paare möglich, in der Regel vertraut man aber nicht unkontrolliert dem Geschick des Anwenders.

Diese Form von Wissen ist aber von einem Angreifer, der sich unbefugten Zugang verschaffen will, noch leichter zu erraten als ein völlig freies Passwort, da durch die Frage der mögliche Wertebereich der Antworten stark eingeschränkt ist. Beschränkt man die Anzahl der möglichen falschen Antworten nicht, kann man mit sog. Wörterbuchangriffen[1] oder durch gezielte Recherche die Antwort, bzw. das Passwort, erraten. Diese Wörterbücher enthalten gängige Passwörter naiver Benutzer (Wohnorte, Mädchen und Jungennamen, Geburtstage, Sportarten, Spitznamen und Helden aus Online-Spielen). Zugleich lassen sich automatisiert daraus Variationen bilden, etwa gespiegelte Begriffe und solche, in denen einzelne Ziffern durch das gewählte Wort wandern.

In der Literatur, wie [Sch06a], [Sch07c], [Sch06b] oder [Gri06], finden sich zahlreiche Hinweise auf die Wahl eines sicheren Passworts, das möglichst aus Groß- und Kleinbuchstaben, Ziffern und gängigen Sonderzeichen bestehen sollte. Offensichtlich wäre ein völlig zufällig gewählte Kombination dieser Zeichen (etwa eine Pseudozufallszahl[2]) optimal, allerdings sind diese Werte vom Menschen nicht merkbar, so dass

[1] Methode, um mit Hilfe einer Passwortliste ein unbekanntes Passwort (oder Benutzernamen) zu knacken.

[2] Als „sicher“ wird in der Praxis eine Erratenswahrscheinlichkeit von kleiner als 0,000 001 genommen, also von 1:einer Million. Dies erreicht man unter Ausschluß einiger auffälliger Kombinationen

Anwender dazu neigen, sich diese Werte irgendwo zu notieren, wodurch der Sicherheitseffekt entfällt. Zusätzlich erschwert die von vielen Organisationen betriebene Strategie, Anwender periodisch (z. B. alle 6 Monate) zum Wechsel der Passwörter zu zwingen und die generelle Empfehlung, nicht für alle Anwendungen das gleiche Passwort zu wählen, die Auswahl guter Passwörter.

Zurück zu Ali Baba. Aus heutiger Sicht war das für den Schatz gewählte Losungswort zu einfach, zumal „Sesam“ im syrisch-persischem Sprachraum als Synonym für Reichtum gilt.

## 1.2 Besitz

Bleiben wir bei Ali Baba. Am Anfang der Geschichte hat er das Passwort zufällig erfahren. Jeder, der gelauscht hätte, hätte es auch erfahren können. Das ist einer der gravierenden Nachteile. Ein erlauschtes Passwort kann von jedem anderen eingesetzt werden. Das ist auch die Gefahr der sogenannten Key-Logger oder wenn naive Anwender Passwörter per email austauschen.

Es liegt daher nahe, Passwörter so geschützt auf eine Datenträger aufzubringen (Chipkarte, USB-Stick), dass sie im Klartext nie auftreten (mehr hierzu später) und auch so mit einer nicht fälschbaren Identität des Datenträgers zu verbinden, dass Duplikate nicht angefertigt werden können. Statt Duplikaten hätten wir auch Nachschlüssel sagen können und in der Tat ist diese Form der Authentifizierung gleichbedeutend mit Schlüsseln, Urkunden, Ausweisen, bzw. dem im Mittelalter üblichen Siegel. Digitale Signaturen, die wir unten auch kurz erwähnen wollen, gehen ebenfalls in diese Richtung, erlauben Sie doch die Authentifizierung des Absenders einer Nachricht und können die Verfälschungsfreiheit der Urkunde garantieren.

Andersherum haben Schlüssel (oder generell die Authentifizierung über Besitz) den Nachteil, dass ein möglicher Dieb (wie bei der Erlauschung des Passworts) mit ihnen ohne weitere Überprüfung alle Rechte des Schlüsselinhabers ausüben kann. Dies gilt für reale Schlüssel (und Türcodekarten, usw.), wie auch digitale Signaturen, deren öffentliche Ungültigkeitserklärung eine aufwändige und damit teure Angelegenheit ist, was die weitere Verbreitung gemäß Signaturgesetz [Roß96] stark behindert.

## 1.3 Biometrie

Wie schon zuvor angemerkt, erlauben Identifikatoren auf der Grundlage von Wissen oder Besitz keine verläßliche Authentifizierung der Person, die den Identifikator präsentiert. Das ist bei der Biometrie anders. Sie beschäftigt sich mit der Messung und Bewertung von biologischen Charakteristika. Der Einsatz der biometrischen Daten in der Authentifizierung von Personen erlebte in den letzten Jahren einen rasant ansteigenden Trend.

ungefähr mit 6 Ziffern wie man es von TANs im Online-Banking kennt. PINs, die man sich merken muß, sind dagegen in der Regel nur vierstellig und damit inhärent unsicher.

Ihr Einsatz zur Authentifizierung wird in der letzten Zeit immer mehr gefordert, zugleich sind die Erfolge bezogen auf die beiden aus dem Information Retrieval abgeleiteten, voneinander unabhängigen Maßzahlen Precision[3] und Recall[4] nicht überzeugend, d.h. es gibt zu viele „false hits“ und zugleich „missed hits“. Bekannte biometrische Merkmale sind die für Reisepässe neuerdings herangezogenen Fingerabdrücke und der Scan der Iris. Beide gelten als für eine Person einzigartig und lassen sich automatisiert auswerten. Prinzipiell wäre im übrigen auch eine Online-DNA-Analyse ein geeignetes Verfahren, allerdings erscheint dies momentan nicht praktikabel.

Eigentlich wäre aber die Identifikation durch „Ansehen“ der Person die natürlichste Form der Authentifikation. Der Augenschein ist offensichtlich die in Alltagssituationen gebräuchlichste Methode, d.h. man gleicht die Erinnerung (Physionomie, Körperbau, Bewegung) mit dem Gegenüber ab. Dabei kann Erinnerung sich auf ein früheres Treffen, ein früher gesehenes Bild oder, etwa bei Personen des öffentlichen Lebens, früher gesehenen Aufnahmen aus Filmen im Kino oder Fernsehen beziehen.

Ist die Person unbekannt, gibt es also keine mental gespeicherte Erinnerung, kann natürlich ein Ausweis herangezogen werden. Dies ist aber keine Biometrie im engeren Sinn, vielmehr steht hier wieder der Besitz zusammen mit der Fälschungssicherheit des Ausweises im Vordergrund. Auf die alternative Methode, der Person ein gesichert identifiziertes Bild aus anderen Quellen gegenüberzustellen, gehen wir in dieser Arbeit ein.

Die Funktionsweise sowie ein paar Vorteile und Nachteile der bekanntesten und meist eingesetzten Authentifizierungsverfahren werden im nächsten Kapitel behandelt.

[3]Unter Precision wird der Anteil der korrekten Suchergebnisse im Verhältnis zur Gesamtzahl der Suchergebnisse verstanden.

[4]Die Recall Rate beschreibt die Anzahl der korrekten Suchergebnisse im Verhältnis der erwarteten korrekten Treffer.

# Kapitel 2

# Stand der Technik

Die oben ausgeführten Verfahren kommen heutzutage selten einzeln zum Einsatz. Es wurden verschiedene Kombinationen entwickelt, um die Sicherheit dieser Systeme zu verbessern. Je nach Einsatzbereich wurden auch vereinfachte Varianten entwickelt.

In diesem Kapitel werden einige einfache Verfahren mit ihren Vor- und Nachteilen vorgestellt, sowie Möglichkeiten, sie zu kombinieren, näher untersucht.

## 2.1 Wissen

### 2.1.1 PIN

Die PIN (Persönliche Identifikationsnummer) ist eine einfache Form des Kennwortes mit einer ausschließlich numerischen Zeichenfolge. Sie ist meistens vier bis sechs Stellen lang und wird sowohl in sicherheitsrelevanten Bereichen eingesetzt, wie bei EC-Karten oder dem Online-Banking, als auch bei nicht sicherheitsrelevanten Bereichen, wie dem Einschalten von Handys.

### 2.1.2 TAN, iTAN und mTAN

Eine TAN (Transaktionsnummer) ist auch eine numerische Form des Kennwortes, welche vorwiegend im Online-Banking verwendet wird. Die TAN besteht üblicherweise aus sechs Dezimalziffern.

Transaktionsnummern sind, wie der Name andeutet, an einen Geschäftsvorfall gebunden und werden „verbraucht“. Daher muss sich der Anwender eine TAN nicht merken, was er auch aufgrund der genannten Länge von üblicherweise 6 Ziffern gar nicht könnte. TANs (speziell die unten erläuterten iTANs) sind wegen ihrer Einmalverwendung den Token ähnlich, die zeit- oder ereignisgesteuert vom Anwender ad-hoc generiert und dem System, gegenüber dem sich der Anwender authentifizieren will, präsentiert werden. Das System erwartet ein spezielles Token, das wie die TAN im Prinzip eine Pseudozufallszahl ist, und kann über die korrekte Angabe die Identität verifizieren. In jedem Fall gehört diese Authentifizierungsmethode in den Bereich des Wissens (nur der Berechtigte kennt die Zahl), auch wenn in der Umsetzung der Besitz einer TAN-Liste oder der Besitz eines Geräts zur Token-Generierung [Vera] notwendig ist, um sich das Wissen zu verschaffen.

Das Generieren, Prüfen und zum Benutzer Übertragen der TAN stellt die größte Herausforderung für die Entwickler dar.
Die Wahrscheinlichkeit, eine sechs-stellige TAN zu raten beträgt 1:1.000.000. Der Benutzer bekommt beispielsweise eine Liste von 100 TANs und hat drei Versuche bei Falscheingabe. Da ergibt sich eine Wahrscheinlichkeit von:
$\frac{100}{10^6} + \left(1 - \frac{100}{10^6}\right) \times \left(\frac{100}{10^6-1} + \left(1 - \frac{100}{10^6-1}\right) \times \frac{100}{10^6-2}\right) \approx \frac{3}{10^4} = 0,03\%$ für das Erraten einer TAN aus der Liste.

Um diese Wahrscheinlichkeit senken zu können, besonders aber auch um einzelne gestohlene TANs (vgl. Phishing in Abschnitt 3.5) wertlos zu machen, werden indizierte TANs (iTANs) verwendet. Der Benutzer kann keine beliebige TAN aus der Liste eingeben, sondern es wird eine bestimmte durch eine Positionsnummer gekennzeichnete angefordert. Ein Betrüger kann diese mit einer Wahrscheinlichkeit von $1 : 1.000.000 = 0.0001\%$ bei einer sechs-stelligen TAN und einem einzigen Versuch erraten. Die iTANs waren auch als Lösung gegen Phishing (Abschnitt 3.5) gedacht.

Die TAN- bzw. iTAN-Liste kann unter Umständen in die falschen Händen geraten, als Lösung entstand die mTAN (Mobile TAN). Diese wird dem Kunden für einen einzigen Vorgang per SMS zugeschickt und ist nur für diesen Vorgang verwendbar. Die Wahrscheinlichkeit geraten zu werden ist bei der mTAN genau so groß wie bei der iTAN. Das besondere Sicherheitsmerkmal der mTANs ist demnach die Verwendung eines zweiten Kommunikationskanals. Unter der Annahme, das die berechtigte Person auch im Besitz eines Handys mit vorab von der Person angegebener Telefonnummer ist, kann ihm so eine TAN zur einmaligen Eingabe übermittelt werden. Verständlicherweise wird diese Methode auch zur Aktivierung von Online-Zugängen der Mobilfunkbetreiber verwandt.

Ein weiteres Verfahren zum Generieren von TANs wurde 2006 von der Baden-Württembergischen Bank eingeführt. Temporäre TANs (jetzt auch von anderen Banken unter dem Namen smartTANplus vermarktet) entsprechen den oben genannten ereignisgenerierten Token und verwenden ein kleines Gerät, den Tokengenerator. Das besondere daran ist die Eingabe der Kontonummer des Zahlungsempfängers in den Generator, wodurch die dann erzeugte TAN nur für Überweisungen an diese Kontonummer gültig bleibt, eine Umlenkung an einen anderen Zahlungsempfänger, der die TAN erlauscht hat, also unmöglich ist. Hinweise zu TANs und weiteren Varianten finden sich u.a. in [Wikb].

### 2.1.3 Challenge-Response

Grundsätzlich kann man Authentifizierungsverfahren, die über das spezielles Wissen eines Teilnehmers die Identität verifizieren, als Challenge-Response-Verfahren (deutsch etwa Herausforderung-Antwort) bezeichnen. Die Herausforderung besteht darin, ein Passwort oder eine PIN, den Mädchennamen der Mutter, oder ein anderes, eigentlich nur dem Teilnehmer bekanntes Wissen zu nennen. Die Antwort ist dann das gerade geforderte Wort.

Häufig verwendet man den Begriff Challenge-Response aber in einem engeren Sinn. Hintergrund ist die Beobachtung, dass der Response, also das geforderte Wort, nicht

unbedingt an den Verifizierer geschickt werden muss, um seien Gültigkeit zu prüfen. Es würde ja genügen, sicher zu wissen, dass der Teilnehmer das geforderte Passwort oder die PIN oder TAN tatsächlich kennt. Hierzu kann man kryptographische Verfahren einsetzen, etwa eine symmetrische Verschlüsselung. Dazu sendet der Verifizierer $\mathcal{A}$ einen zufälligen Text an Teilnehmer $\mathcal{B}$. Dies bezeichnet man jetzt als Challenge. Teilnehmer $\mathcal{B}$ wendet lokal bei sich ein Transformationsverfahren (in der Regel eine Verschlüsselung) unter Eingabe des geheimen Wissens (des Schlüssels) an und erzielt daraus eine Nachricht (Response), die an den Verifizierer $\mathcal{A}$ geht. Dieser muss nun ein Verfahren haben, mit dem er aus dem Response zweifelsfrei schließen kann, dass $\mathcal{B}$ das Geheimnis (meist den Schlüssel) kennt. Damit wäre die Authentifizierung gelungen.

Vorwiegend wird die Nachricht mit einem symmetrischen Verschlüsselungsverfahren (Kapitel 4) verschlüsselt. Der Benutzer $\mathcal{A}$ generiert eine zufällige Datenmenge $\mathcal{D}$. Sendet sie $\mathcal{B}$ zu, der sie mit einem Schlüßel $\mathcal{K}$ verschlüsselt $(B.ENC_K(\mathcal{D}))$ und die Antwort an $\mathcal{A}$ zurückschickt. Gleichzeitig verschlüsselt $\mathcal{A}$ die Datenmenge $(A.ENC_K(\mathcal{D}))$ und vergleicht das Ergebnis mit dem von $\mathcal{B}$ (Abbildung 2.1, Seite 8). Der Einsatz der asymmetrischen Verschlüsselung (Kapitel 4) ist auch möglich.

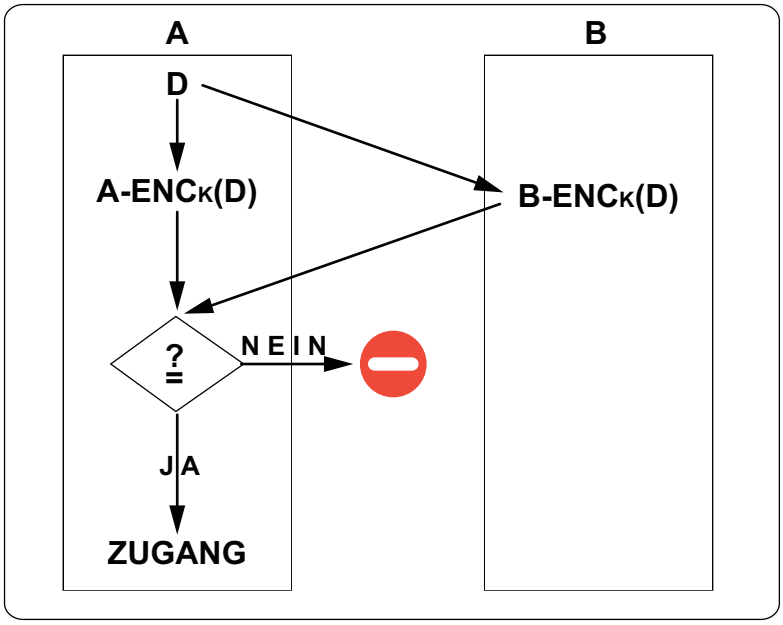



Abbildung 2.1: Challenge-Response-Verfahren

## 2.2 Besitz

Bei Authentifizierung über Besitz denkt man im Falle der Zugangsberechtigung meist an den Türschlüssel, den man am Schlüsselbund in der Tasche trägt. Für die Übertragung in die Informationstechnik tritt heute meist eine Karte, deren auslesbarer digitaler Inhalt als Identifikator dient, an die Stelle des physischen Schlüssels. Genauso bildet der Besitz von EC- und Kreditkarten heute die Grundlage des bargeldlosen Zahlungsverkehrs und auch Ausweise werden heute vorzugsweise im „Scheckkartenformat“ ausgestellt. Es folgt eine kurze Übersicht heutiger Techniken.

### 2.2.1 Magnetkarte

1972 wurde die Magnetkarte von Hewlett Packard als Speichermedium für Taschenrechner eingeführt. Danach wurde sie auch von anderen Herstellern wie Texas Instruments übernommen.

Heutzutage wird sie vorwiegend zum Speichern der Authentifizierungsdaten eingesetzt und dient zum Beispiel als Mitgliedskarte bei vielen Vereinen oder als Guthabenträger in Parkplätzen oder für Kopiergeräte. Im Bankwesen wird diese Technologie in den Debitkarten (ec-Karten) und Kreditkarten verwendet. Die Magnetkarten sind mit einem Magnetstreifen versehen, der von einem speziellen Kopf gelesen und geschrieben wird. Um Missbrauch verhindern zu können, kann man die Karte mit einem weiteren Schutz erweitern, indem der Benutzer bei der Verwendung der Magnetkarte sich noch authentisieren muss. Dies geschieht meist durch die Eingabe einer Geheimzahl (PIN) oder durch eine Unterschrift.

### 2.2.2 Chipkarte

Die Chipkarte ist eine Kunststoffkarte, meist aus PVC hergestellt, mit eingebautem Chip (Abbildung 2.2, Seite 9). Sie können auch einen Mikroprozessor enthalten, der außer dem Speichern von Daten weitere Funktionen anbietet, wie deren Ver- bzw. Entschlüsselung (Abbildung 2.3, Seite 10). Diese werden oft auch als Smartcard bezeichnet.

Die Chipkarten sind in drei Größen zu finden (nach der ISO-7816 Norm):

**85,60 x 53,98 mm:** Das am weitesten verbreitete Format. Wird bei einigen Debitkarten, Telefonkarten, dem EU-Führerschein oder der Gesundheitskarte verwendet

**66 x 33 mm:** Wird am wenigsten verwendet

**25 x 15 mm:** Findet vor allem bei SIM-Karten in Mobiltelefonen Verwendung

Auf einigen speziellen Mikroprozessor-Karten ist ein reduziertes Betriebssystem installiert, das das Ausführen von vom Benutzer selbst programmierten Funktionalitäten ermöglicht. Die weit verbreitetste davon ist die Javakarte mit dem Java-Betriebssystem.

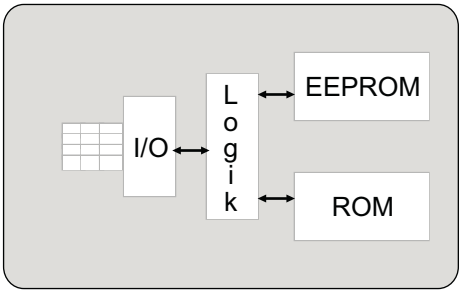



Abbildung 2.2: Standard Chipkarte

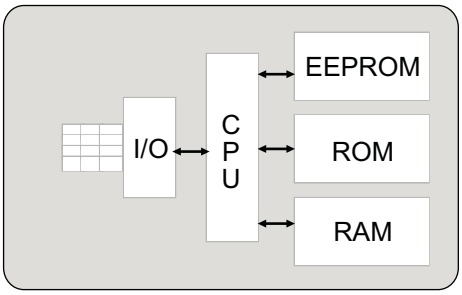



Abbildung 2.3: Mikroprozessor-Chipkarte

Die Spezialkarten können in der Regel weniger Funktionen ausführen als die allgemeinen Karten, diese dafür aber schneller, weil sie bei der Entwicklung für die gewünschten Funktionalitäten optimiert wurden.

### 2.2.3 RFID

Radio Frequency Identification RFID (ins Deutsche übersetzt: Identifizierung über Radiowellen) ist ein Verfahren zur kontaktlosen Übertragung der Authentifizierungsdaten.

Ein RFID-System besteht aus 3 Grundkomponenten: einem Transponder $\mathcal{T}$, einem Lesegerät ($\mathcal{L}$) und dem EDV-System. Die Übertragung geschieht über Radiowellen $\mathcal{RW}$ (Abbildung 2.4, Seite 10)

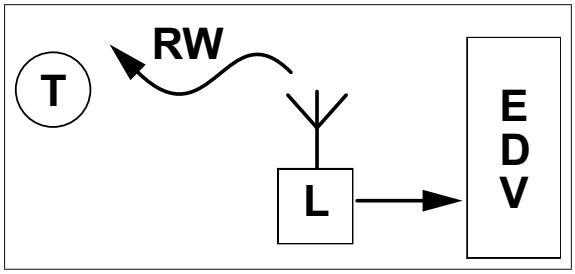



Abbildung 2.4: RFID

Prinzipiell funktioniert die RFID-Kommunikation folgendermaßen: Das Lesegerät erzeugt ein elektromagnetisches Wechselfeld mit einer bestimmten Frequenz. Der Empfang dieses Signales durch die Transponder-Antenne generiert einen Induktionsstrom, welcher den eingebauten Mikrochip aktiviert. In dem vom Lesegerät ausgesendeten Feld sind Befehle moduliert, deren Antwort vom Mikrochip wiederum in das Feld moduliert wird.

Die genaue Funktionsweise des RFID-Systems kann [wika] entnommen werden.

RFIDs werden heute in Europa bereits Tieren, speziell Hunden, eingepflanzt und sind Vorschrift für den grenzüberschreitenden Reiseverkehr. In diesem Fall, wie in allen anderen Fällen oben, in denen zunächst nur ein Identifikatorwert geliefert wird,

muß es noch eine Verifikationsmöglichkeit geben, den Identifikatorwert mit der Entität in Beziehung zu setzen. Beim „gechipten“ Hund ist es ein Tierausweis, dessen Besitz - zusammen mit dem am Tier ausgelesenen RFID - das Tier authentifiziert. In den meisten anderen Fällen wird über einen Server der gelieferte Identifikatorwert in Bezug zu Kontonummern und anderen Entitätsattributen gesetzt um festzustellen, dass dies ein „passender“ Schlüssel ist.

Zuletzt sei darauf hingewiesen, dass Authentifikation per Besitz immer voraussetzt, dass das Besitzgut nicht leicht kopierbar ist, oder dass eine Kopie ohne weiteres Wissen nutzlos ist. Dies ist für physische Schließsysteme oft leichter durchzusetzen als für die elektronischen Äquivalente, wenn Rohlinge frei verfügbar sind, auf die geschickte Fälscher die Daten des Orginals kopieren können.

## 2.3 Biometrie

### 2.3.1 Fingerabdruckerkennung

Der Fingerabdruck eines Menschen, auch Fingerbild genannt, ist individuell. Selbst eineiige Zwillinge haben unterschiedliche Fingerbilder. Anhand dieses Merkmals kann man eine Person eindeutig identifizieren. Der Einsatz des Fingerabdrucks zur Authentifizierung war aber mit enormem Aufwand verbunden.

Die Elektronisierung dieses Verfahrens gilt als Meilenstein in der Entwicklung der Authentifizierungssystemen.

Bei dieser Methode hat man sich auf Merkmale des Fingerbildes beschränkt, wie Gabelungen, Schleifen und Wirbel, auch Minutien genannt, die während des ganzen Lebens unverändert bleiben.

Um die Fingerbilderkennung durchzuführen, wird der Finger zunächst durch spezielle Sensoren aufgenommen. Daraus entsteht ein Bild der Papillarlinien des Fingers in Graustufen.

Durch Erhöhung des Kontrastes, Entfernung von Bildstörungen und weitere Bildbearbeitungsfilter lassen sich die Minutien aus dem Bild besser extrahieren. Dieser Vorgang wird in der Fachsprache als Normierung des Bildes bezeichnet.

Aus dem normierten Bild werden die Minutien mit ihrer Position eingezeichnet und abgespeichert und werden zum Vergleich bei Bedarf eingesetzt.

Aus einem Fingerabdruck können bis zu 100 Minutien abgelesen werden. Da nur 14 davon für eine sichere Identifikation ausreichen, bleibt viel Spielraum, falls der Finger beim Einscannen leicht verdreht oder verschoben aufgelegt wird. (Abbildung 2.5, Seite 12)


Abbildung 2.5: Die Fingerabdruckerkennung

### 2.3.2 Gesichtserkennung

Dieses biometrische Verfahren lässt sich mit dem Auge vergleichen. Das Gesicht wird vom Auge aufgenommen und ins Gehirn übertragen, das das Bild analysiert. Je nachdem ob die Identifikation oder die Verifikation der Zweck der Erkennung ist, wird das analysierte Bild mit dem vorgelegten oder mit dem im Gehirn abgespeicherten verglichen. Im Fall einer Übereinstimmung geht man davon aus, dass die Person die ist, für die sie sich ausgibt (Verifikation), oder dass die Person aus mehreren identifiziert ist (Identifikation).

Ein System zu entwickeln, das alle Merkmale des Gesichts vergleichen soll, wäre undenkbar. Trotz der schnellen Rechenanlagen und günstigen Resourcen, die einem heutzutage zur Verfügung gestellt werden können, wäre so ein Vergleich sehr zeitaufwändig. Deswegen wird die Gesichtserkennung nur auf einige Merkmale begrenzt.
Die Gesichtserkennung läuft nach diesen Schritten ab (Abbildung 2.6, Seite 13):

1. **Gesichtsdetektion:** Es wird geprüft, ob das hochauflösende Bild ein Gesicht enthält.
2. **Gesichtslokalisierung:** Es wird festgestellt, wo sich das Gesicht auf dem Bild befindet.
3. **Lokalisierung der Augen:** Auf dem Gesicht werden die Augen lokalisiert.
4. **Normierung des Gesichts:** Die Größe und Orientierung des Bildes werden angepasst, damit es der Norm entspricht. Dabei wird häufig von der Position der Augen ausgegangen.
5. **Berechnung der Merkmale:** Aus dem Bild werden weitere Merkmale extrahiert (Datensatz zum Vergleich).
6. **Berechnung der Ähnlichkeit:** Die extrahierten Daten werden mit den Referenzdaten verglichen und die Übereinstimmungen berechnet.
7. **Vergleich mit Schwellenwert:** Mit dem Schwellenwert wird festgestellt, wie viele Übereinstimmungen ausreichend wären, um eine Person zu identifizieren bzw. zu verifizieren. Je nachdem wie sicher das System sein soll.

8. **Entscheidung:** Liegt die Übereinstimmung über einem vorher festgelegten Schwellenwert, so gilt das Gesicht als erkannt, liegt sie unter diesem Wert, dann erfolgt eine Zurückweisung.

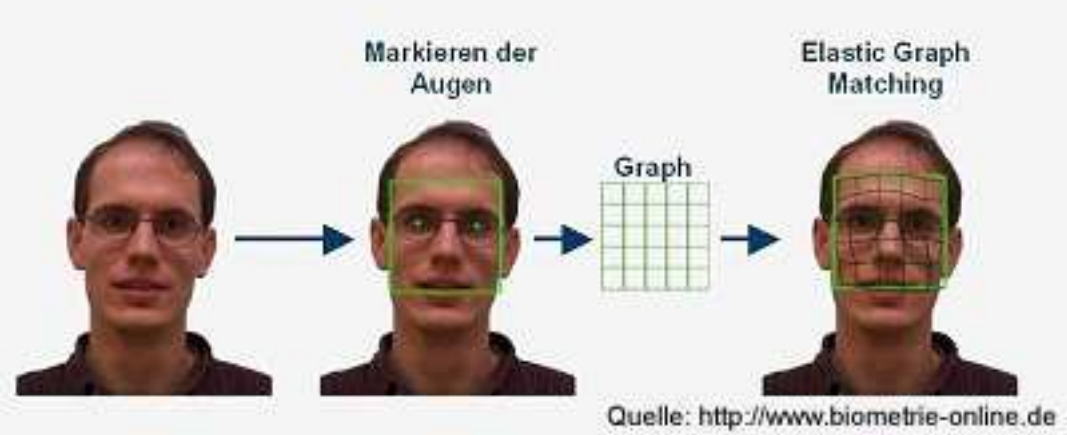



Abbildung 2.6: Die Gesichtserkennung

### 2.3.3 Iriserkennung

Die Iris, auch Regenbogenhaut genannt, ist die gefärbte Blende des Auges. Sie trennt die vordere von der hinteren Augenkammer und reguliert den Lichteinfall. Die Struktur der Iris bildet sich in den ersten Lebensmonaten und bleibt über die restliche Lebenszeit weitgehend unverändert. Es sind bis jetzt keine Fälle bekannt, in denen die Irisstruktur bei zwei Personen identisch ist. Selbst die des rechten und linken Auges einer Person haben unterschiedliche Strukturen. Somit ist die Iris sehr gut zur Identifizierung einer Person geeignet.

In der Biometrie sind heutzutage über 260 Merkmale der Iris analysierbar, und können jeweils mehrere Werte haben. Die Wahrscheinlichkeit, zwei Iris mit einer 70-prozentigen Übereinstimmung zu finden, liegt bei eins zu sieben Milliarden [KBG].

Die meisten kommerziellen Iriserkennungssysteme setzen den Algorithmus von John Daugman ein, dessen Basis die Verwendung von Gaborwavelets bildet und in folgende Schritte unterteilt werden kann:

1. Lokalisierung des Mittelpunktes der Iris
2. Feststellung der Radien der Irisränder
3. Transformation des Irisringes in eine Darstellung bezüglich eines pseudopolaren Koordinatensystems
4. Berechnung des Iriscodes durch 2D-Gaborwavelets

Zusammenfassend lässt sich das Daugman-Prinzip folgendermaßen beschreiben [KBG]:

Die Aufnahme des Augenbildes erfolgt mit einer (digitalen) schwarzweiß Kamera. Um möglichst viele Informationen zu erhalten, verwenden die Kameras neben dem sichtbaren auch infrarotes Licht. So können auch die sehr dunklen Irismerkmale sichtbar gemacht werden. Nach der Aufnahme des Bildes erfolgt zuerst die Trennung der Iris von den Augenliedern, der Pupille und dem Augenweiß. (Abbildung 2.7, Seite 14)
Die Iris wird mittels einer Schablone unterteilt. Um das Muster der

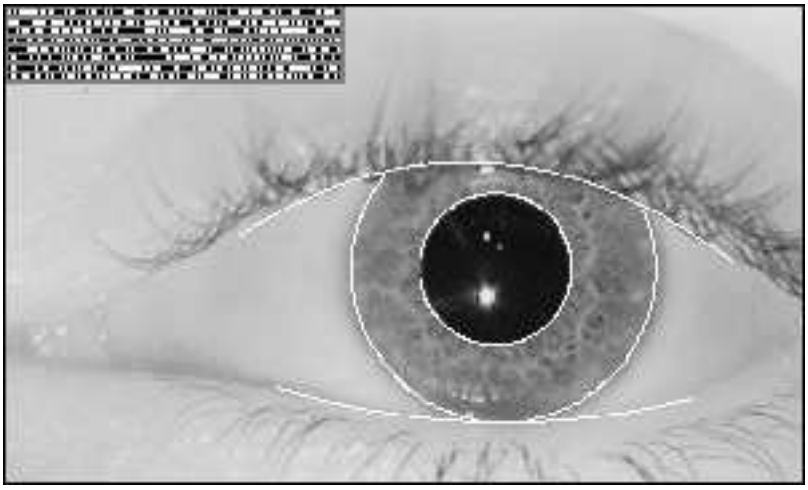


Abbildung 2.7: Die Iriserkennung

Iris zu bestimmen, werden nun die Schattierungen zwischen den hellen und den dunklen Regionen der Iris in einen Code umgewandelt. Dies geschieht durch komplexe Berechnungen, wobei unter anderem die Wavelet-Transformation zum Einsatz kommt. Der entstandene Iriscode kann in einer Datenbank gespeichert werden und steht so für den Gebrauch eines Iriserkennungssystems zur Verfügung .

## 2.4 2/3-Factor- vs. Multipart-Authentication

In den kritischen Fällen, in denen ein Höchstmaß an Sicherheit gefordert wird, reicht es meistens nicht aus, eine der oben genannten Verfahren einzusetzen. Wenn man den Zugang mit anderen Voraussetzungen verbinden will, zum Beispiel eine Tür die sich nur öffnen lässt wenn mehrere Personen anwesend sind, oder der Zugriff auf eine Datei, der erst erlaubt ist, wenn der Besitzer der Datei auch anwesend ist usw., kommt die so genannte „2/3-Factor- oder Multipart-Authentication“ zum Einsatz.

Diese Authentifikation basiert auf mehreren Verfahren, welche gleicher oder verschiedener Art sein können.

Bei der „Multipart-Authentication“ benötigt man eine erfolgreiche Authentifizierung mehrerer oder aller Teilnehmer. Dieses Verfahren wird zum Beispiel in manchen Groupware-Lösungen eingesetzt. Um zu verhindern, dass ein einziger Administrator das Passwort eines Benutzers ändern kann und sich somit Zugang zu dessen Daten verschaffen kann, ist die Zustimmung weiterer Administratoren nötig.

Auf der [Sec] ist die „Two-Factor-Authentication“ wie folgt erklärt:

> Bei einer Two-Factor-Authentication werden mindestens zwei unterschiedliche Faktoren für eine erfolgreiche Authentifikation benötigt. Im Falle von Security Tokens (z.B. von Kobil, RSA, Safeword etc.) sind dies z.B. ein kleines Gerät, welches nur einmal gültige Passwörter generiert und eine PIN. Der eine Faktor ist die Hardware, welche die Codes generiert. Ohne den Besitz der Hardware kann man sich nicht einloggen. Der zweite Faktor ist das Wissen der korrekten PIN Nummer. Nur wenn beide Faktoren vorhanden sind, kann sich der Benutzer erfolgreich authentifizieren.

Analog dazu werden drei Faktoren bei der „3-Factor-Authentication“ benötigt.

# Kapitel 3

# Angriffsmethoden

Die im Kapitel 2 genannten Verfahren dienen dazu, unberechtigten Personen den Zugang zu geheimen Bereichen oder Daten zu erschweren. Angreifer werden trotzdem versuchen die Informationen auszuspähen.
Diese Angriffsmethoden sind das Thema dieses Kapitels.

## 3.1 Raten

Raten eines Passworts ist die älteste und uneffektivste Methode, sich einen unbefugten Zugang zu verschaffen. Der Angreifer sollte den Passwort-Besitzer gut kennen und ahnen, was er sich als Passwort ausgedacht hat.

Eine begrenzte Anzahl an Eingabeversuche und eine intelligente Passwortgenerierung, d.h. eine Kombination aus Ziffern, Klein-, Großbuchstaben und Sonderzeichen und Verzicht auf die Nutzung der allgemein bekannten Daten wie Vor-, Nachname, Geburtsdaten oder ähnlichem, machen das Raten des Passwortes nahezu unmöglich.

Eine erweiterte Variante des Ratens, in der Kenntnisse über eine Person nicht nötig sind, ist der Wörterbuchangriff. Man verwendet eine Liste (mit Inhalten eines Wörterbuchs) der als Passwort naiver Teilnehmern meist benutzten Begriffe und probiert sie einzeln durch, bis man das richtige findet. Die oben genannten Tipps machen auch dieses Verfahren wirkungslos.

## 3.2 Brute-Force-Attack

Ken Thompson, einer der Väter von UNIX, wird mit dem Satz „When in doubt, use brute force“ zitiert.

Die „Brute-Force-Attack“ kann wortwörtlich als „Angriff mit roher Gewalt“ übersetzt werden. Bei dieser Angriffsmethode sind keine Kenntnisse über das Passwort bzw. den Besitzer notwendig, es werden einfach alle mögliche Kombinationen ausprobiert.

Eine „Brute-Force-Attack“ ist einfach zu implementieren, der Rechenaufwand steigt aber exponentiell mir der Länge des gesuchten Begriffes. Für ein vier-stelliges Wort bestehend aus Klein- und Großbuchstaben ohne Ziffern ergeben sich zum Beispiel bereits $(2 * 26)^4 = 52^4 = 7.311.616$ mögliche Kombinationen.

Passwörter sind oft als Hash-Werte (verschlüsselt, s. Kapitel 4) gespeichert, unter UNIX früher sogar für alle sichtbar. Aus dem Hashwert kann nicht das Passwort (der Klartext) berechnet werden, der Zugang erfolgt nur durch Präsentation des korrekten Passworts im Klartext. Ist die Hash-Funktion (der Verschlüsselungsalgorithmus) bekannt, was meist der Fall ist, und kennt man den gespeicherten Hash-Wert des Passworts, kann man in einem Brute-Force-Angriff solange Passwörter generieren und verschlüsseln, bis man eine Übereinstimmung mit dem Hash-Wert des gespeicherten Passworts findet und damit das Passwort als Klartext kennt. Auf einem handelsüblichen Computer können mehrere hunderttausend Passwörter pro Sekunde ausprobiert werden. Somit sind längere und aus einer größeren Anzahl an verwendeten Zeichen bestehende Passwörter für einen ausreichenden Schutz gegen den „Brute-Force-Angriff“ erforderlich.

Man kann den Wörterbuchangriff als eine intelligentere Form des „Brute-Force-Angriffs“ betrachten. Es werden Listen mit den meist benutzten Passwörter generiert [Sch06b]. Die auszuprobierenden Passwörter sind dann auf die Einträge dieser begrenzten Liste eingeschränkt.

## 3.3 Trojaner, Würmer und Viren

Ein Trojanisches Pferd, auch Trojaner genannt, ist ein Programm, das als nützliche Anwendung getarnt, im Hintergrund heimlich andere Aktivitäten ausführt. Der Begriff leitet sich aus der griechischen Mythologie ab. Bekanntlich gewannen die Griechen den Trojanischen Krieg mit einer Kriegslist, indem sie den belagerten Bewohnern Trojas ein hölzernes Pferd als Geschenk vor die Tore stellten, in dessen Inneren Soldaten verborgen waren. Diese öffneten in der Nacht die Tore, so dass die Griechen die Stadt einnehmen konnten.

Grundsätzlich müssen die Aktivitäten des Trojaners nicht schädlich sein. Meist werden sie im Alltag zum Ausspionieren genutzt. Sie werden auf den Computer des Anwenders eingeschleust und als nützliches Programm getarnt. Im Hintergrund übertragen sie jedoch die geheimen Daten des Opfers oder installieren weitere schädliche Viren oder Würmer, die eigenständig und unabhängig vom Trojaner laufen können. Zuletzt sind Trojaner unter dem ironisch gemeinten Begriff „Bundestrojaner“ als Einschleusungsmethode für die vom Gesetzgeber angedachte Online-Durchsuchung von IT-Systemen bekannt geworden.

Im Gegensatz zum Trojaner ist ein Computervirus ein Programm, das sich selbst auf einem Computer verbreitet. Er infiziert vorhandene Dateien und kann durch diese Dateien zu anderen Systemen übertragen werden. Ein Virus muss vom Benutzer ausgeführt werden, meistens durch Anklicken der infizierten Programme, damit er sich verbreitet. Eine eigenständige Verbreitung ist nicht möglich.

Das ist der wichtigste Unterschied zu einem Computerwurm. Würmer warten nicht passiv darauf, mit infizierten Dateien weitergegeben zu werden, sondern verbreiten sich selbstständig über Computernetzwerke. Sie verbreiten sich zum Beispiel durch das Versenden infizierter E-Mails (selbstständig durch eine SMTP-Engine oder durch ein E-Mail-Programm), durch IRC-, Peer-to-Peer- und Instant-Messaging-Programme oder über Dateifreigaben. Dieser Missbrauch der Netzwerkresourcen sowohl auf dem infizierten als auch auf dem zu infizierenden Computer, richtet große wirtschaftliche Schäden an.

Trojaner, Viren und Würmer richten dreierlei Schäden an. Zum einen verfolgen sie meist kriminelle Ziele, wie Manipulation von lokalen Dateien oder Einstellungen, Übertragung geheimer Daten, manchmal auch gezieltes Lahmlegen von IT-Systemen. Die direkten Schäden bei Erfolg sind offensichtlich, etwa das Ausspähen von Passwörtern oder Industriegeheimnissen. Zweitens muss bei erfolgreichem Eindringen, auch wenn sie keinen direkten Schaden angerichtet haben, das System von den Eindringlingen befreit werden, was häufig mit einer Neuinstallation, Stillstand bzw. Nichtverfügbarkeit bei Diensten, Datenverlust usw. verbunden ist. Drittens kostet auch die erfolgreiche Abwehr vor dem Eindringen in das System bares Geld durch wiederholte Plattenscans, Bezug von Abwehrprogrammen (Antivirenprogrammen, die aber natürlich auch Trojaner und Würmer bekämpfen) und deren permanentes Update. Zuletzt sind viele Anwendungsprogramme komplexer, als es ihre Funktionalität eigentlich erfordert, nur um auf mögliche Angriffe der genannten Schädlinge eingerichtet zu sein. Beispiele sind email-Programme mit Virenschutz und Internetbrowser allgemein.

## 3.4 Spyware, Backdoor, Keylogger und Sniffer

Als Spyware bezeichnet man ein Programm, welches sich über einen Wurm, Virus oder trojanisches Pferd auf einem Rechner installiert und so programmiert ist, dass es im Hintergrund lauscht und persönliche Daten des Benutzers an den Entwickler des Spywareprogramms oder Dritte sendet. Diese Daten können in harmlosen Fällen zu Werbezwecken genutzt werden, im nicht-harmlosen Fall zum Ausspähen geheimer Daten, die der Besitzer zum Beispiel beim Online-Banking oder ähnlichem eingibt, mit fatalen Konsequenzen.

Ein weiteres Verfahren, das Angreifern das Ausspähen von persönlichen Daten erlauben soll, sind Backdoors, auf deutsch Hintertüren. Backdoors sind auch eine Malware[1], welche dem Entwickler eine Hintertür zu einem System zur Verfügung stellt und ihm das Umgehen der Sicherheitsmaßnahmen ermöglicht.

Der Keylogger ist ein Tastatur-“recorder“, er kann Software oder Hardware sein. Wenn er installiert und aktiviert ist, kann er die Tastaturaktivitäten aufzeichnen und dem Angreifer senden. Aus diesen Aufzeichnungen lassen sich dann eingegebene Daten wie PIN-Nummern oder Passwörter auslesen.

[1]Computerprogramme, welche vom Benutzer unerwünschte, und ggf. schädliche Funktionen ausführen.

Die Software-Keylogger lassen sich auch über Viren, Würmer oder Trojaner verbreiten, die Hardware-Keylogger müssen zwischen Tastatur und Rechner eingebaut werden, deswegen können sie nur eingesetzt werden, wenn der Zugang zur Hardware möglich ist.

Der Sniffer ist eine Software zur Netzwerk-Analyse, die den Datenverkehr empfangen, aufzeichnen, darstellen und auswerten kann. Diese Diagnoseprogramme werden zunehmend von Angreifern zur Datenspionage missbraucht. Die durch Sniffer empfangenen Daten hängen von der Netzwerkstruktur ab. Ist ein Hub eingesetzt, dann kann ein „Schnüffler“ alle Netzwerkdaten analysieren. Wird ein Switch verwendet, ist nur den Datenverkehr zu sehen, der für das sniffende System selbst bestimmt ist. Diesen Schutz zu umgehen, ist den Sniffern mittels ARP-Spoofing [GH02] und Flooding [Verb] inzwischen auch gelungen.

## 3.5 Phishing & Pharming

Der Begriff „Phishing“ ist ein Wortspiel durch Verkürzung des Begriffs „Passwort Fishing“. Gemeint ist das „Abfischen“ von geheimen Zugangsdaten, etwa zum Online-Banking oder für Zahlungssysteme wie PayPal. Der Köder ist eine den Originalseiten des Unternehmens nachgeahmte Seite, die den Empfänger dazu verleiten soll, Passwort, TAN, PIN usw. einzugeben, meist unter dem Vorwand, eine Sicherheitsüberprüfung erfordere dies. Die so vom Angreifer abgefangen und missbraucht werden.

Die Phishing-Nachrichten beinhalten meistens einen Verweis, der dem Richtigen sehr ähnelt, was dem Opfer nicht auffällt, zum Beispiel
*www.meinescherebank.de/...* anstatt
*www.meinesicherebank.de/....*
Der Empfänger öffnet im guten Glauben den Link und wird auf die präparierte Seite geleitet.

Man kann sich vor Phishing schützen, indem man die altbekannte Adresse (www.meinesicherebank.de) manuell eingibt und somit sicherer sein kann[2], auf der originalen Seite zu landen. Vorausgesetzt wird dabei, dass die Seite gegen Einbetten fremder Inhalte, zum Beispiel durch Cross-Site Scripting (XSS) oder Frame-Spoofing [Sch07a], geschützt. Diese Maßnahme kann vor Phishing schützen aber nicht vor Pharming.

Pharming basiert wie Phishing auf der Umlenkung zu gefälschten Seiten. Die Auflösung einer URL (Internetadresse) in eine IP-Adresse geschieht durch eine DNS-Anfrage an einen Domain-Name-Server. Um diesen zu entlasten, werden nur unbekannte URLs angefordert, die bekannten werden vom Betriebssystem in einer Datei zur weiteren Nutzung eingetragen. Gelingt es einem Angreifer, diese Datei zu manipulieren, könnte einer URL eine falsche IP-Adresse zugeordnet werden und somit auf eine beliebige Seite umgeleitet werden. Alternativ kann der Rechner des Opfers so manipuliert werden, dass er anstatt den richtigen Domain-Name-Server, den des

[2] häufig werden aber auch gezielt gängige Tippfehler ausgenutzt.

Angreifers anfragt. Der Angreifer kann dann die gewünschten Adressen austauschen und zu gefälschten Seiten umlenken. Man spricht hier von DNS-Spoofing.

Eine weitere Pharming-Methode ist das „Cache Poisoning“. Der Angreifer bringt gefälschte Daten in den Cache eines Domain-Name-Servers ein. Alle Clients, die diesen Server anfragen, werden auf manipulierte Webseiten gelenkt.

## 3.6 Man-In-The-Middle-Angriff

Bei dem Man-in-the-Middle-Angriff (im Deutschen Mann-in-der-Mitte), auch als Janusangriff bekannt, versucht der Angreifer sein System in ein Rechnernetzwerk einzuschleusen. Dies geschieht physisch oder auch logisch. Die Kommunikation wird dann über den Angreifer geleitet, und ist zum Ausspionieren bzw. zur Manipulation offen.

Die folgende Definition ist dem Glossar des Heise-Verlags entnommen[Hei]:

> Bei diesem Angriff täuscht ein böswilliger Dritter „in der Leitung“ zwei Kommunikationspartner, indem er ihnen jeweils die Identität des anderen vorspiegelt. Typischerweise geschieht das, um eine an sich sichere Verschlüsselung auszuhebeln.
> Beide Kommunikationspartner verschlüsseln zwar ihre Daten, aber so, dass sie der in der Mitte sitzende Angreifer entschlüsseln und wie ein Proxy an den anderen weiterleiten kann. Voraussetzung dafür ist, dass er eine Verbindungsanfrage auf sich umleiten kann und die Kommunikationspartner die Identität des Gegenübers nicht überprüfen. Über MITM-Angriffe lassen sich beispielsweise gesicherte SSL-Verbindungen zum Online-Banking belauschen.

```
Ohne MITM

----------                    ----------
| Client |<================>| Server |
----------       ^            ----------
                 |
        verschlüsselte Verbindung

Mit MITM

----------       ------          ----------
| Client |<=====>|MITM|<========>| Server |
----------       ------          ----------
```

## 3.7 Buffer Overflow

Der Buffer Overflow ist eine der technisch anspruchsvollsten Angriffsmethoden, sie setzt sehr gute Programmierkenntnisse voraus. Der Angreifer nutzt die Program-

mierfehler des Betriebssystems oder einer Applikation aus und kann dadurch seinen Schadcode ausführen lassen.

Zusammengefasst - und ohne auf die Details näher einzugehen - lässt sich das Grundprinzip des Buffer Overflow folgendermaßen erklären:
Sowohl die Daten als auch die Rücksprungadressen werden vom Prozessor in zwei verschiedenen Bereichen des Speichers gespeichert, die aneinander grenzenden Heap- und Stacksegmente (Abbildung 3.1, Seite 21). Gelingt es einem Angreifer, die Rücksprungadresse zu ändern, kann man dann beliebigen Code ausführen lassen und hat damit das Ziel erreicht.

Als erstes muss die Adresse des eigenen Codes im Speicher (im Codesegment) rausgefunden werden. Die Adresse muss dann als Rücksprungadresse eingetragen werden.

Die Rücksprungadresse einer Funktion wird vom System mit den lokalen Variablen in dem Stacksegment abgelegt. Der Angreifer manipuliert das Programm, sodass es einen größeren Pufferinhalt (z.B. 12345678) in einen vorhandenen Puffer (1234) kopiert und dadurch die abgesicherten Daten nach dem Puffer (hier die Daten ABCD) überschreibt, vorausgesetzt die Rücksprungadresse (RS-Adresse) ist direkt nach den Puffer abgelegt.

**Beispiel:**

| Puffer | | | | RS-Adresse | | | |
|---|---|---|---|---|---|---|---|
| Vor der Manipulation: | | | | | | | |
| 1 | 2 | 3 | 4 | A | B | C | D |
| Nach der Manipulation: | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

**Stacksegment:** Lokale Variable und gesicherte Prozessorregister, die das Programm zu einem späteren Zeitpunkt wieder benötigt, unter anderen die Rücksprungadressen.

**Heapsegment:** Vom Programm zur Laufzeit angeforderte Speicherbereiche, zum Beispiel mittels *malloc()*, häufig auch globale Daten und Konstanten.

**Codesegment:** Maschinenbefehle des Programms. In Größe und Inhalt fest und gegen Überschreiben geschützt.

Sowohl der Heap als auch der Stack können zur Laufzeit des Programms wachsen.

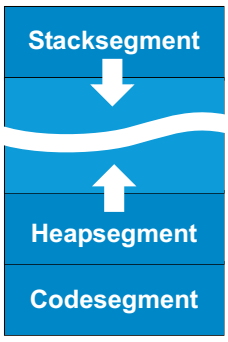



Abbildung 3.1: Hauptspeichersegmente

Derartig manipulierter Speicher führt dazu, dass das in der Abarbeitung befindliche Programm bei der Rückkehr von einer Unterroutine eine andere, als die vorgesehene Rücksprungadresse lädt und an diese Adresse mit dem Start des Schadprogramms verzweigt.

In diesem Kapitel sind nur die häufig eingesetzten Angriffsmethoden aufgeführt. Es sind aber noch weitere Verfahren vorhanden, auf die hier nicht eingegangen wird, weil sie nicht mehr so verbreitet sind, oder weil sie den vorgestellten sehr ähneln.

Heutzutage ist außer eines Schutzes der wichtigen Daten durch moderne Zugriffseinschränkungen auch die Verschlüsselung der Daten erforderlich. (Kapitel 4)

# Kapitel 4

# Verschlüsselung / Kryptographie

So wie eine Bank sich nie hundertprozentig gegen einen Überfall schützen kann, sowenig kann man den Zugang zu wichtigen Daten völlig sichern. Banken versuchen bei einem Überfall dem Dieb eine Farbbombe in die Beute zu schmuggeln, die dann zeitverzögert explodiert und die geraubten Scheine unbrauchbar macht. Das entsprechende Gegenstück in der Datenhaltung, allerdings vorbeugend angewandt, ist die Verschlüsselung. Auch wenn die Dateien in unbefugte Hände gelangen, sind sie nutzlos, da sie den Inhalt im Klartext nicht preisgeben.

Ver- und Entschlüsseln von Botschaften (Informationen) hat eine lange Historie. Die Wissenschaft der Verschlüsselung heißt Kryptographie (Wissenschaft der Geheimschrift), die Kryptologie wiederum ist die Wissenschaft, die sich mit den Verfahren der Informationssicherheit beschäftigt. Sie umfaßt neben der Krypthographie auch die Kryptoanalyse zur Rückgewinnung der Information oder zur Erlangung des verwendeten Schlüssels.

In diesem Kapitel werden ein paar Begriffe erläutert, einige historische Verschlüsselungssysteme werden kurz beschrieben, und wird auf einigen symmetrischen und asymmetrischen Verfahren etwas näher eingegangen.

## 4.1 Einführung

### 4.1.1 Algorithmus vs. Schlüssel

Zu einem Ver- bzw. Entschlüsselungssystem gehören:

**Algorithmus:** besteht aus mathematischen Funktionen, die das Ver- bzw. Entschlüsseln der Daten ermöglichen.

**Schlüssel:** sind zusätzliche Daten, die der Algorithmus zum Ver- bzw. Entschlüsseln benötigt. In manchen Verfahren benötigt man mehr als einen Schlüssel, man spricht dann von einem Schlüsselsatz.

Die Sicherheit eines Verschlüsselungssystems beruht nicht auf das Geheimhalten des Algorithmus, was man in der Fachsprache als „Security through obscurity“ (Sicherheit durch Verschleierung) bezeichnet, sondern auf der Schwere des Erratens des Schlüssels. Dies muss einerseits durch Einsatz von langen Schlüsseln erreicht werden,

andererseits durch die Wahl der richtigen Algorithmen, die mit dem richtigen Schlüssel eine einfache und schnelle Verschlüsselung bieten, ohne Kenntnis des Schlüssels aber die Entschlüsselung praktisch unmöglich machen. Dahinter steckt - außer in den einfachen historischen Fällen - eine mathematische Theorie (bzw. anders gesehen ein mathematisches Problem), aus der sich Funktionen (Lösungen des Problems) mit der gerade genannten Eigenschaft zur Realisierung durch die Kryptoalgorithmen ableiten lassen.

**Das Prinzip Von Kerckhoffs** *Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus abhängen. Die Sicherheit gründet sich nur auf die Geheimhaltung des Schlüssels.*

Versucht ein Angreifer, den Schlüssel mit der „Brute-Force-Methode“ zu knacken, muss er im Schnitt die Hälfte aller möglichen Schlüssel ausprobieren. Die Anzahl der möglichen Schlüssel steigt exponential mit der Länge des Schlüssels. Zum Beispiel können aus einem 64-Bit Code $2^{64} = 1,8 * 10^{16}$ mögliche Schlüssel generiert werden. Bei einem 128-Bit Code sind es dagegen $2^{128} = 3,4 * 10^{38}$ Schlüssel.

Das mathematische Problem, auf dem das Verschlüsselungssystem basiert, muss verhindern, dass ein Schlüssel schon mit geringem Aufwand unbefugt berechnet werden kann.

### 4.1.2 Kodierung

Eine der ältesten Methoden der Verschlüsselung ist die Kodierung. Der Sender und Empfänger der Nachricht haben sich über eine Kodierungstabelle geeinigt, welche Worte oder Sätze einem Code zuordnet. Zum Beispiel „10:30“ bedeutet „Auftrag ist erfüllt“ und „11:30“ bedeutet „Auftrag nicht erfüllt“. Mit einem Satz wie „ich kann erst um 10:30 Uhr kommen“ kann der eine dem anderen den Erfolg des Auftrages mitteilen.

Das Problem bei den Codes ist, dass sie bei mehrfachem Gebrauch an Sicherheit verlieren, da ein Spion die Zuordnung erkennen könnte.

Ein anderes Problem der Kodierung ist, dass die Tabelle beschränkt ist. Man bräuchte neue Tabellen, wenn neue Nachrichten hinzugefügt werden müssen, was ein neues Verständigen des Kommunikationspartner voraussetzt. Die Lösung für das Problem ist die Chiffrierung, wodurch jeder beliebige Text chiffriert werden kann.

### 4.1.3 Chiffrierung

Bei der Chiffrierung werden Zeichen aus der Nachricht mit einem eindeutigen Zeichen oder Symbolen ersetzt. Jedem Zeichen wird immer das selbe, eindeutige, Zeichen zugeordnet, daher die Bezeichnung in der Fachsprache als „monoalphabetische Chiffrierung“.

Die Chiffrierung kann mit geringem Aufwand umgegangen werden, sie ist als Verschlüsselungsmethode nicht zu empfehlen. Die Chiffrierung wird meistens eingesetzt, wenn ein Text nicht, unbeabsichtigt, gelesen werden soll, zum Beispiel die Lösung eines Quiz, d.h. eine Frage wird gestellt, die richtige Antwort wird chiffriert und der Schlüssel wird verraten. Will einer die Antwort erfahren, kann er sie mit wenigem Aufwand dechiffrieren. Wie und warum die „monoalphabetische Chiffrierung“ unsicher ist kann [Beu96] entnommen werden.

#### 4.1.3.1 Caesar-Verschlüsselung, ROT13

Bei der Caesar-Verschlüsselung wird jeder Buchstabe des Alphabets um eine bestimmte Zahl rotiert. Zum Beispiel ROT13.
ROT13, eine der aller einfachsten und unsichersten Ver- bzw Entschlüsselungsmethoden, wird zum Beispiel von einigen Mail-Programmen geboten, um die Nachrichten zu verschlüsseln. Diese Technik „rotiert“ die Buchstaben der Nachricht um 13 Stellen, d.h. aus einem A wird ein N usw.
**Beispiel:**
*Das ist eine geheime Nachricht* mit ROT13 verschlüsselt ergibt
*Qnf vfg riar grurvzr Anpuevpug.*

Man kann diese Technik verbessern, wenn man das Vertauschen der Zeichen zufälliger macht. Man generiert zufällige Zeichenketten, in denen jedes Zeichen nur einmal vorkommt und benutzt sie zum Vertauschen als Schlüssel.
**mögliche Zeichen:** abcdefghijklmnopqrst
**Schlüssel:** bhdzne734dm6ury9i1gk
**Nachricht:** das ist eine geheime nachricht
**verschlüsselt:** zbg 4gk n4rn 7n3n4un rbd314d3k

Man kann das Knacken dieser Methode noch weiter erschweren, indem man zum Beispiel verschiedene Schlüssel generiert und für jede Textzeile der Reihe nach einsetzt.

#### 4.1.3.2 Freemason'sche Chiffrierung

Freemason hat die Buchstaben durch Symbole ersetzt. (Abbildung 4.1, Seite 25)

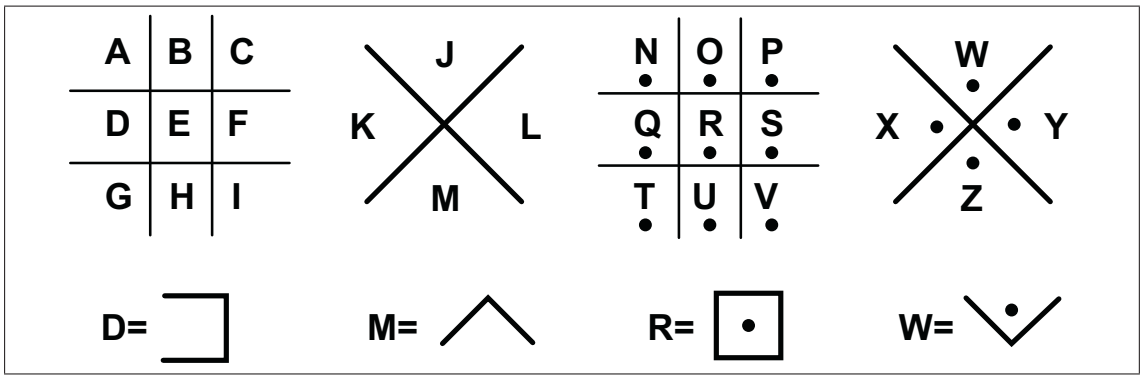



Abbildung 4.1: Beispiel der Freemason'schen Chiffrierung

#### 4.1.3.3 One-Time Pads

Die Methode der One-Time Pads ist eine Bündelung von Kodierung und Chiffrierung, basierend auf der Idee eines „Notizblocks“. Jede Seite dieses Blocks ist eine Kodierungstabelle, welche nur einmal verwendet werden darf. Die Kommunikationspartner müssen identische Blöcke haben. Nach Gebrauch einer Seite muss sie aus dem Block entfernt werden.

Die modernen auf One-Time Pads basierenden Systeme setzen keine Papierblätter ein, sondern eine Reihe von Zahlen. Jede Zahl entspricht einem Zeichen der zu verschlüsselnden Nachricht und gibt an, um wieviele Stellen es rotiert werden soll.
**Beispiel:**
Die Zahlenliste lautet 01 23 07 14.
Die Nachricht *Hallo Leute* soll verschlüsselt werden
H wird um 01 Stelle rotiert und ergibt I,
a um 23 Stellen ergibt x,
l um 07 Stellen ergibt s,
l um 14 Stellen ergibt z,
o um 01 Stelle ergibt p,
␣ um 23 Stellen ergibt 7,
L um 07 Stellen ergibt S,
usw...

Eine weitere Variante dieses Verfahrens ist „Vernam-One-Time-Pad“. Anstatt einer Rotation des Zeichens, wird eine bitweise XOR-Operation mit der Zahl aus dem „Pad“ durchgeführt.

Die „Vernam-One-Time-Pad“-Methode ist theoritisch sehr sicher, stellt sich aber in der Praxis als ineffizient heraus, weil viele und lange Schlüssel ausgetauscht werden müssen.

### 4.1.4 Blockchiffrierung, Stromchiffrierung

Bei der Blockchiffrierung wird die Nachricht in Blöcke gleicher Größe aufgeteilt und die Blöcke werden verschlüsselt.

Bei der Stromchiffrierung wird die Verschlüsselung bitweise durchgeführt.

## 4.2 Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung wird auch Geheimer-Schlüssel-Kryptographie genannt. Man bezeichnet sie als symmetrisch, weil derselbe Schlüssel sowohl für die Verschlüsselung als auch für die Entschlüsselung verwendet wird oder der Entschlüsselungsschlüssel aus dem Verschlüsselungsschlüssel leicht zu berechnen ist. Wer den Schlüssel besitzt, kann die Daten chiffrieren bzw. dechiffrieren, deswegen muss der Schlüssel gut und sicher aufbewahrt werden. Daher auch die Bezeichnung „geheimer Schlüssel“.

Um eine sichere Kommunikation zu erreichen, muss jedem Kommunikationspaar Sender-Empfänger, die zugleich Empfänger-Sender sind, ein Schlüssel zugeordnet werden, der auch regelmäßig geändert werden soll. Die Schlüssel müssen verwaltet werden, damit eine aktuelle und gültige Schüssel-Paar-Zuordnung gewährleistet werden kann.

Außerdem ist eine verschlüsselte Kommunikation nur möglich, wenn man der Gegenseite den Schlüssel zukommen lassen hat, zum Beispiel durch das Verfahren „Diffie-Hellman-Schlüsselaustausch“ [Ali05].

Die Anzahl der benötigten Schlüssel bei paarweise unterschiedlichen Schlüsseln ergibt sich aus:

$$(n) \cdot (n-1) \div 2$$

wobei $n$ die Anzahl der Personen ist. Zum Beispiel zwei Personen benötigen einen gemeinsamen Schlüssel, drei Personen drei und fünf Personen benötigen zehn Schlüssel. (Abbildung 4.2, Seite 27)

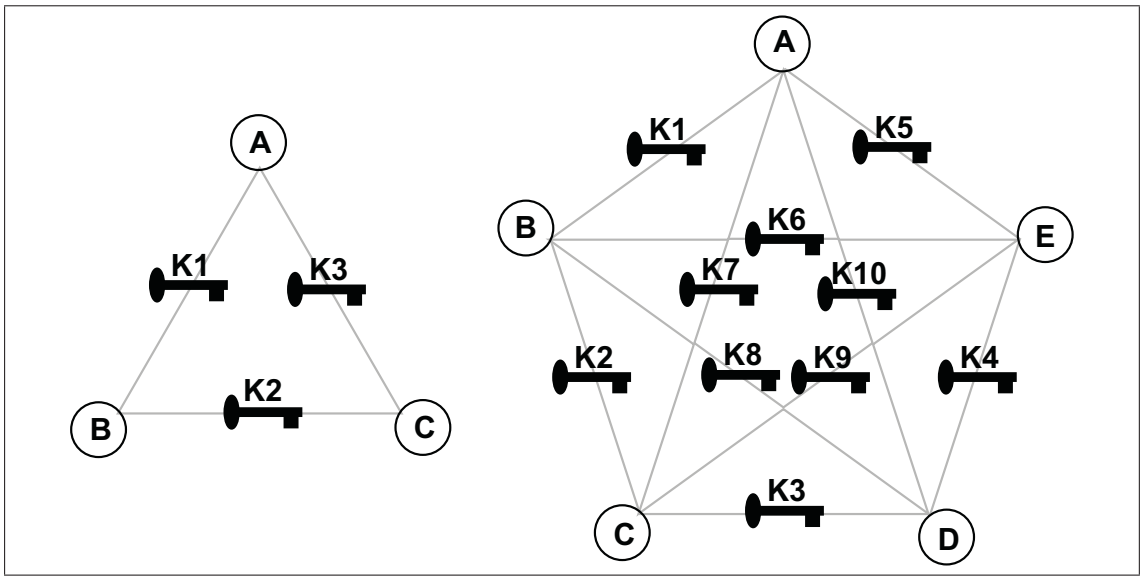



Abbildung 4.2: Symmetrische-Kryptographie

Hier ein paar Beispiele der symmetrischen Verschlüsselung Algorithmen:

**DES:** Data Encryption Standard (Veröffentlicht 1975) ist ein Blockchiffrierungsalgorithmus mit einer Blockgröße von 64 Bits. Das heißt die Daten werden in 64 Bits-großen Blöcke geteilt, jeder Block Klartext wird in einen 64-Bit-Block Chiffriertext verschlüsselt. Die Daten werden nach dem Schema von Feistel [Buc01] bearbeitet und mit einem Schlüssel der Länge 64 Bits verschlüsselt. Jedoch werden von diesen 64 Bits nur 56 Bits zur Ver- bzw. Entschlüsselung verwendet, die übrigen 8 Bits (jeweils ein Bit aus jedem Byte) werden zum Paritäts-Check benötigt. Man spricht von einem 64 Bits langen Schlüssel und einer effektiven Schlüssellänge von 56 Bits.

**Triple-DES:** Auch 3DES oder TDES genannt, ist eine dreifache DES-Verschlüsselung mit verschiedenen Schlüsseln. Die Schlüssellänge von Triple-DES ist mit 168 Bits dreimal so groß wie bei DES (56 Bits), wodurch die Schlüsselkomplexität um den Faktor $2^{112}$ gesteigert wird. Die effektive Schlüssellänge liegt aber nur bei 112 Bits. Es ist aber relativ langsam, da der Rechenaufwand durch die dreimalige Verschlüsselung hoch ist. Dabei wird jeder Datenblock mit einem DES-Schlüssel $K_1$ chiffriert, dann mit $K_2$ dechiffriert und mit $K_3$ chiffriert
$3\mathrm{DES}_{(K_1,K_2,K_3)} := \mathrm{DES}_{(K_1)} \circ \mathrm{DES}^{-1}_{(K_2)} \circ \mathrm{DES}_{(K_3)}$
Andere Kombinationen aus drei DES-Verschlüsselungen sind möglich.

**IDEA:** Bei dem International Data Encryption Algorithm (Veröffentlicht 1991) ist die Chiffrierung blockbasiert. Das Verfahren benutzt einen 128 Bits langen Schlüssel und eine Blockgröße von 64 Bits. Der Schlüssel wird in 16 Bits Teilschlüssel unterteilt. Jeder 64-bit Klartextblock wird in vier 16-bit-Teilblöcke zerlegt. Die Teilblöcke werden mehrfach mit den Teilschlüsseln verschlüsselt, dabei werden drei Operationen durchgeführt, die logische Operation $XOR$, die Addition modulo $2^{16}$ und die Multiplikation modulo $2^{16}+1$. [Ewa05] erläutert den Vorgang ausführlicher.

**AES:** Advanced Encryption Standard (Veröffentlicht 2000) ist das Ergebnis eines Wettbewerbs, zu welchem Anfang 1997 das US-amerikanische National Institute of Standards and Technology (NIST) aufgerufen hat. Der Algorithmus (Sieger) wurde von Rijndael entwickelt. AES besitzt eine Blockgröße von 128 Bit und eine variable Schlüssellänge von 128, 192 oder 256 Bit. Anhand der Schlüssellänge wird zwischen den drei AES-Varianten AES-128, AES-192 und AES-256 unterschieden. Jeder Block wird in 16, 8 Bits große, Teilblöcke geteilt, welche in den Zellen einer $4 \times 4$ Tabelle geschrieben werden. Dann wird nacheinander bestimmten Transformationen unterzogen. Der Klartextblock wird nacheinander mit verschiedenen Teilen des Schlüssels verschlüsselt, was man als Runde bezeichnet. Die Anzahl dieser Runden variiert und ist von der Schlüssellänge abhängig. [DR99] behandelt AES genauer.

Es sind weitere symmetrische Verfahren verfügbar, wie u.a. Rivest Cipher #2, #4, #5 und #6, FEAL und CAST. Sie werden in dieser Arbeit nicht erläutert, da die o.g. Verfahren nur als Beispiel der symmetrischen Kryptosystemen gedacht sind.

## 4.3 Asymmetrische Verschlüsselung

Im Jahr 1970 wurde die asymmetrische Verschlüsselung, auch Öffentlicher-Schlüssel-Kryptographie genannt, entwickelt und stellte die traditionelle Kryptographie auf den Kopf. Im Gegensatz zur symmetrischen Verschlüsselung, in der die Kommunikationspartner denselben Schlüssel zum Ver- bzw. Entschlüsseln verwenden, werden bei der asymmetrischen zwei zu einander passende Schlüssel erzeugt, einen zum Verschlüsseln, den anderen zum Entschlüsseln.

Der Schlüssel zum Chiffrieren kann jedem zur Verfügung gestellt werden, daher die Bezeichnung „öffentlicher Schlüssel“. Er kann auf direktem Weg, auf öffentlichen

Web-Seiten oder von „Key Distribution Center“ (KDC)[1] veröffentlicht werden. Er ermöglicht nur die Verschlüsselung. Nur wer den passenden Dechiffrierschlüssel besitzt, kann die Daten entschlüsseln, deswegen ist er gut aufzubewahren und keinem freizugeben, darum wird er „privater Schlüssel“ genannt.

Will eine Person $B$ einer Person $A$ eine verschlüsselte Nachricht zukommen lassen, dann muss sich $B$ den öffentlichen Schlüssel von $A$, kurz $P_A$, beschaffen, mit welchem $B$ die Nachricht verschlüsseln kann. Wer den privaten Schlüssel von $A$, kurz $S_A$, hat, also prinzipiell nur die Person $A$, kann die Nachricht entschlüsseln. (Abbildung 4.3, Seite 29)

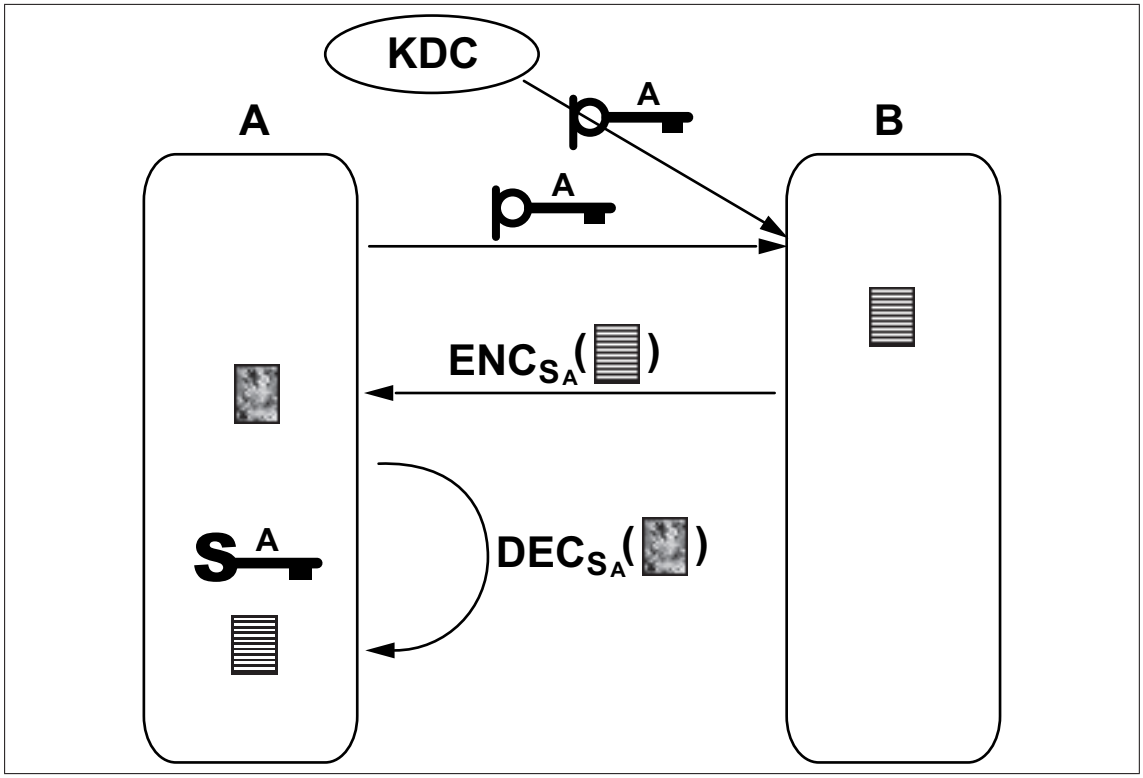



Abbildung 4.3: Öffentlicher-Schlüssel-Kryptographie

Die Schlüssel bei der asymmetrischen Verschlüsselung zeichnen sich durch eine entscheidende Eigenschaft aus. Aus der Kenntnis des öffentlichen Schlüssels ist der private Schlüssel nicht zu erschließen. Mit einer mathematischen Funktion kann ein Klartext unter Verwendung des öffentlichen Schlüssels einfach verschlüsselt werden, der verschlüsselte Text darf aber nicht einfach zu entschlüsseln sein, wenn man nicht über den privaten Schlüssel verfügt. Um das zu erreichen, werden in den asymmetrischen Systemen die sogenannte Einwegfunktionen benötigt. Diese sind in [Beu96] folgendermaßen definiert:

> Dies sind Funktionen, die „Klartexte“ in „Geheimtexte“ transformieren, aber nicht invertiert werden können. Mit anderen Worten: Eine Einwegfunktion ist eine umkehrbare Funktion mit der (unglaublichen!) Eigenschaft, dass es zu jedem gegebenen „Geheimtext“ nur mit unvorstellbar

[1] Eine Institution, die die öffentlichen Schlüssel verwaltet und anderen zur Verfügung stellt.

> großem Aufwand möglich ist, sein Urbild zu berechnen.
> [· · · ]Tatsächlich sind die meisten zeitabhängigen Vorgänge Einwegfunktionen; solche Vorgänge können in einer Richtung äußerst leicht durchgeführt werden, sind aber praktisch nicht umkehrbar: Sie werden zweifellos schon festgestellt haben, dass die Menschen in Ihrem Bekanntenkreis immer älter werden, aber keiner jünger.

Mathematisch kann man nicht beweisen, dass eine Funktion eine Einwegfunktion ist. In der Praxis werden bekannte mathematische Probleme eingesetzt, die nach dem aktuellen Stand der Wissenschaft unlösbar sind, wie das „Faktorisierungs-“, das „Diskrete Logarithmen-“ oder das „Quadratwurzel“-Problem [Beu96]. Um solche Funktionen umkehrbar zu machen, sind zusätzliche Daten nötig, sie werden als privater Schlüssel eingesetzt. Näheres dazu im nächsten Abschnitt.

Ein weiteres Problem, das die asymmetrische Verschlüsselung mit sich bringt, ist die Verwaltung und die Authentizität der Schlüssel. Man muss sicher sein, dass der angebotene öffentliche Schlüssel auch der Person gehört, mit der man kommunizieren will. Deswegen müssen die KDCs unbedingt vertrauenswürdig sein.

### 4.3.1 Asymmetrische Verschlüsselungs-Systeme

Anhand einiger Verfahren wird in diesem Abschnitt erläutert, wie sich die Entwickler von Kryptosystemen (Algorithmen) mathematische Probleme zu Nutze gemacht haben, um Daten zu ver- bzw. entschlüsseln.

#### 4.3.1.1 RSA

RSA (1977 veröffentlicht) ist nach seinen Entwicklern R. Rivest, A. Shamir und L. Adleman benannt und das meist eingesetzte asymmetrische Verschlüsselungssystem.

In diesem System wird das Faktorisierungsproblem eingesetzt. Man benötigt zwei zufällige Primzahlen derselben Größenordnung zum Generieren der Schlüssel. Je größer die Zahlen sind, desto sicherer sind die Schlüssel[2].

Das RSA Verfahren geht folgendermaßen vor:

- Zwei (große) zufällige Primzahlen $p$ und $q$
- $k = p \cdot q$ berechnen
- $m = (p-1) \cdot (q-1)$ berechnen
- Zufällige Zahl $s$ sodass $1 < s < k$ und $ggT(s, m) = 1$
- $t$ finden sodass $s \cdot t \; mod \; m = 1$

Das Paar $(s, k)$ bildet den öffentlichen Schlüssel. Der private Schlüssel besteht aus $(t, k)$. Man beachte, dass aus den öffentlichen Daten $(s, k)$ den privaten Schlüssel nicht berechnet werde kann.

[2] Es gibt weitere Kriterien, die bei der Wahl der Zahlen beachtet werden müssen.[Buc01, S. 117ff.]

Zum Verschlüsseln wird ein Wert $G$ (Geheimtext) aus dem Klartext $T$ berechnet
$G = T^s \ mod \ k$
Der Geheimtext $G$ ermöglicht auch nicht das Berechnen des privaten Schlüssels (Faktorisierungsproblem).

Zum Entschlüsseln wird der private Schlüssel $(t, k)$ benötigt. Aus $G$ wird folgendermaßen $T$ berechnet.
$T = G^t \ mod \ k$

Mit einem nummerischen Beispiel wird die Funktionsweise deutlicher. Für die nächsten Verfahren wird auf das Beispiel wegen der Ähnlichkeit verzichtet.
$p = 43, q = 67$
$k = 43 \cdot 67 = 2881$
$m = 42 \cdot 66 = 2772$
$s = 125$
$t = 377$
Der zu verschlüsselnde Text „JA“ wird nummerisch beispielsweise so dargestellt $T = 1001$. Dann kann der Geheimtext mit dem öffentlichen Schlüssel $(125, 2881)$ so berechnet werden:
$G = 1001^{125} \ mod \ 2881 = 147$
Aus dem Geheimtext $G = 147$ lässt sich der Klartext mit dem privaten Schlüssel $(377, 2881)$ wieder berechnen
$T = 147^{377} \ mod \ 2881 = 1001$

Interessant ist, wie unregelmäßig die Zuordnung „Klartext“⇒„Geheimtext“ aussieht. Beispiele:
$1001 \Rightarrow 147, 1002 \Rightarrow 483, 1003 \Rightarrow 1932, 1004 \Rightarrow 66, 1005 \Rightarrow 938, ...$

#### 4.3.1.2 Rabin

Das Rabin-Verschlüsselungssystem (1979 veröffentlicht) basiert auch auf zwei Primzahlen $p$ und $q$. Der Privat-Schlüssel ist das Paar $(p, q)$, der öffentliche Schlüssel ist das Produkt $n = p \cdot q$.
Zum Verschlüsseln wird aus dem Klartext $T$ der chiffrierte $G$ wie folgt berechnet $G = T^2 mod \ n$.
Um die Entschlüsselung zu vereinfachen, werden die Zahlen $p$ und $q$ so gewählt, dass $p \equiv q \equiv 3 \ mod \ 4$. Dann lassen sich die vier Quadratwurzeln der $G \ modulo \ n$ berechnen, aus denen eine den entschlüsselten Text darstellt. Dafür berechnet man $r$, $s$, $x$ und $y$.

- Mit dem erweiterten euklidischen Algorithmus werden zwei ganze Zahlen $a$ und $b$, mit $ap + bq = 1$, bestimmt.
- $r = G^{(p+1)/4} mod \ p$
- $s = G^{(q+1)/4} mod \ q$
- $x = (aps + bqr) \ mod \ n$
- $y = (aps - bqr) \ mod \ n$

- Die vier Quadratwurzeln von $G$ *modulo* $n$ sind: $x$, $-x \mod n$, $y$ und $-y \mod n$.

Die Entschlüsselung liefert zusätzlich zum Klartext drei weitere Ergebnisse, das richtige Ergebnis muss daher erraten werden. Dies ist der große Nachteil des Rabin-Kryptosystems.
Die bis jetzt eingesetzte Lösung für dieses Problem ist eine bestimmte Struktur des originalen Klartextes (wie die Wiederholung der letzten $i$ Bits), die den Nutztext von den Redundanzen unterscheidet.

#### 4.3.1.3 ElGamal

Die ElGamal-Verschlüsselung (veröffentlicht 1985) basiert auf dem mathematischen Problem der Berechnung des Diskreten Logarithmus. Es wird eine große Primzahl $p$ benötigt, sodass $p-1$ einen großen Primfaktor $q$ besitzt, eine Primitivwurzel $g \mod p$ der Ordnung $p-1$ und ein zufälliges $0 \leq a \leq p-2$.
Das Tripel $(p, g, A)$ kann als öffentlicher Schlüssel eingesetzt werden, wobei $A = g^a \mod p$. Der geheime Schlüssel entspricht $a$.

Zur Verschlüsselung des Klartextes $T$ $(0 \leq T \leq p-1)$ geht man folgendermaßen vor. Man wähle eine zufällige Zahl $1 \leq b \leq p-2$ und berechne
$B = g^b \mod p$ und
$G = A^b T \mod p$
Der chiffrierte Text besteht aus dem Paar $(B, G)$.

Mit dem geheimen Schlüssel $a$ lässt sich der verschlüsselte Geheimtext $(B, G)$ zum Klartext $T$ entschlüsseln. Dazu berechnet man $T = B^{p-1-a}G \mod p$.

#### 4.3.1.4 Elliptische Kurven

Elliptische-Kurven-Verschlüsselung (ECC - Elliptic Curve Cryptography) stellt eine attraktive Alternative zum RSA-Algorithmus dar. Das dem ECC-Verfahren zugrundeliegende mathematische Problem ist die Berechnung des diskreten Logarithmus (DL) in geeigneten Mengen.
Die hier betrachtete Menge besteht aus Punkten, die eine mathematische Gleichung erfüllen, und zwar die einer elliptischen Kurve $y^2 + x^3 + ax + b$. (Abbildung 4.4, Seite 33)
Der entscheidende Vorteil dieses Verfahrens besteht darin, dass man mit deutlich geringeren Schlüssel- und Parameterlängen auskommt, ohne Abstriche hinsichtlich der Sicherheit in Kauf nehmen zu müssen. Es genügen die x-Koordinate eines Punktes sowie ein weiteres Bit, um einen beliebigen Punkt eindeutig identifizieren zu können. Eine ausführlichere Einführung ist in [Cer07] zu finden.

#### 4.3.1.5 Vergleich zwischen RSA und ECC

Die Tabelle 4.1 (Seite 34) stellt die die Entwicklung der RSA- und ECC-Verfahren im Laufe der Jahre zusammen.

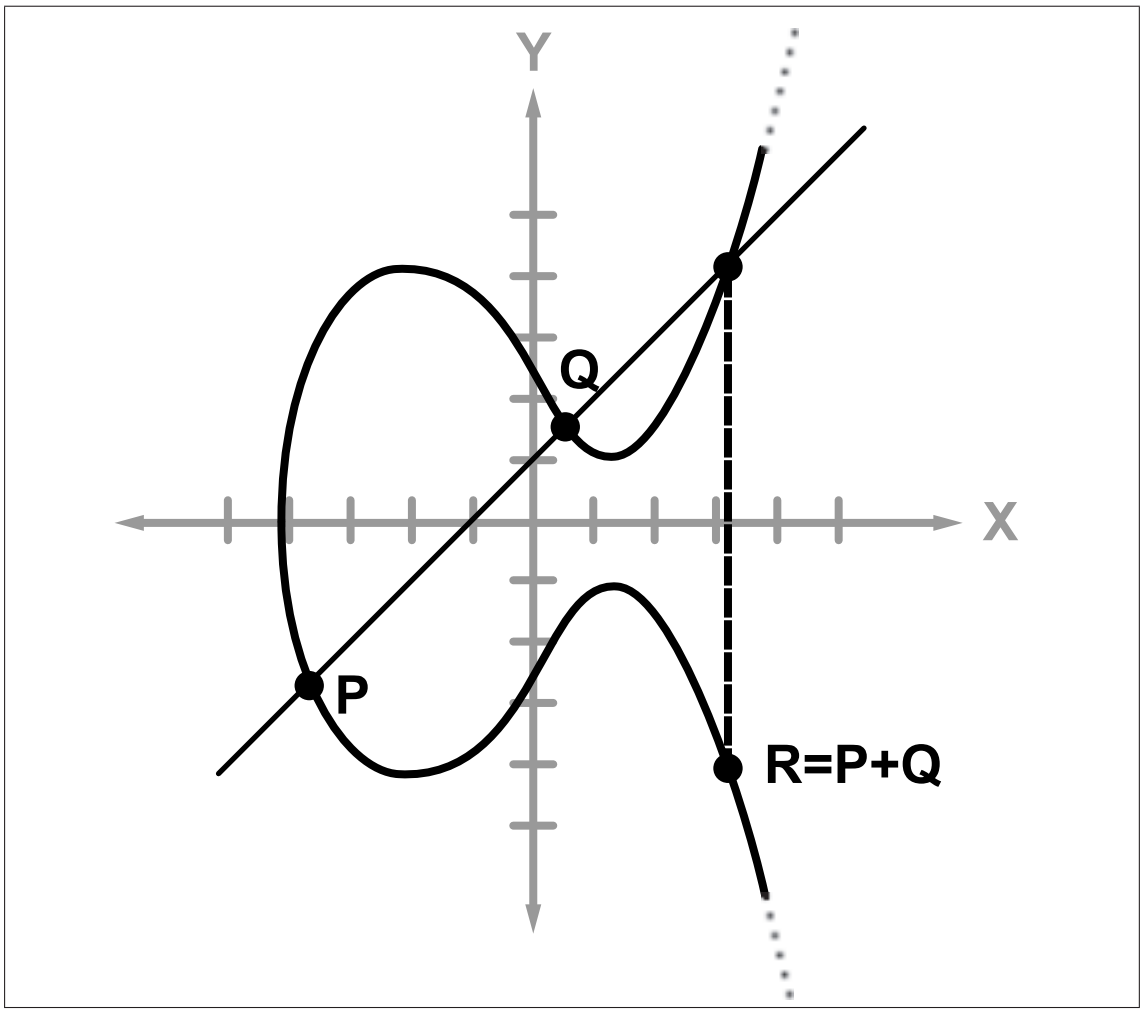



Abbildung 4.4: Grafische Darstellung der elliptischen Kurven

#### 4.3.1.6 ID-basierte Verschlüsselung

Die Identität-basierte Verschlüsselung ist ein Sonderfall der asymmetrischen Verschlüsselung. Der öffentliche Schlüssel besteht nicht aus zufälligen Daten sondern aus personenbezogenen Informationen, die ohnehin allen bekannt sein dürfen und die Person eindeutig identifizieren.

Eine vertrauenswürdige Institution (trusted authority) $T$ generiert den zu dieser Information passenden privaten Schlüssel $S$. Dabei soll ausschließlich $T$ einen gültigen Schlüssel generieren können. Dafür werden Daten benötigt, die nur $T$ bekannt sind. In der Praxis benutzt $T$ einen privaten Schlüssel.

Da die Information die Person eindeutig identifizieren soll, ist bei dem Verfahren die Authentifizierung der Person beim Ausstellen des Schlüssels nicht nötig.

Das Ziel dieses Verfahrens ist es, zu ermöglichen, nur durch Kenntnis bekannter Fakten, zum Beispiel des Namens einer Person, ihr eine verschlüsselte Nachricht schicken zu können. Man bräuchte dann die öffentlichen Schlüsseln nicht mehr auszutauschen.(Abbildung 4.5, Seite 34)

| Jahr | Schlüssellänge symmetrischer Verfahren | Asymmetrische Schlüssellänge (z.B. RSA) | Schlüssellängen von ECC | Erforderliche Jahre auf 450 Mhz PC |
|---|---|---|---|---|
| 2000 | 70 | 952 | 132 | $1,58 * 10^7$ |
| 2002 | 72 | 1028 | 137 | $4,59 * 10^7$ |
| 2004 | 73 | 1108 | 141 | $1,33 * 10^8$ |
| 2006 | 75 | 1191 | 145 | $3,84 * 10^8$ |
| 2008 | 76 | 1279 | 149 | $1,11 * 10^9$ |
| 2010 | 78 | 1369 | 153 | $3,22 * 10^9$ |
| 2012 | 80 | 1464 | 157 | $9,32 * 10^9$ |
| 2014 | 81 | 1562 | 162 | $2,70 * 10^{10}$ |
| 2016 | 83 | 1664 | 166 | $7,81 * 10^{10}$ |
| 2018 | 84 | 1771 | 170 | $2,26 * 10^{11}$ |
| 2020 | 86 | 1881 | 175 | $6,54 * 10^{11}$ |

Quelle: www.cryptovision.com

Tabelle 4.1: Vergleich zwischen RSA und ECC

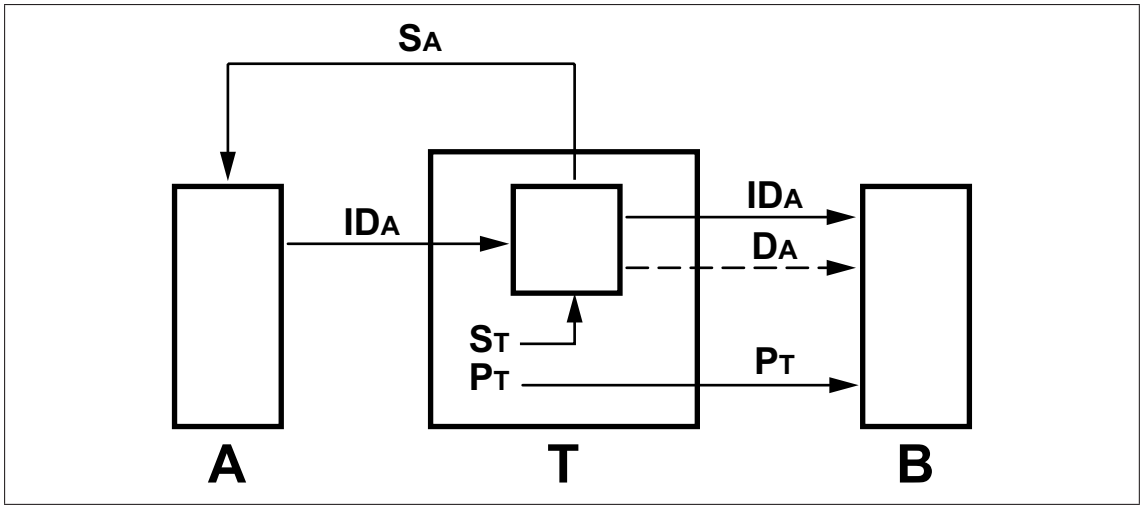



Abbildung 4.5: Generierung des ID-basierten Schlüssels

In einigen Systemen werden noch zusätzliche personenbezogene systemabhängige Daten $D$ generiert, die veröffentlicht werden sollen. Man spricht von „nicht reinen“ ID-basierten Systemen.

#### 4.3.1.7 Hybrid-Verschlüsselung

Der Rechenaufwand bei der asymmetrischen Verschlüsselung und die damit verbundene geringe Geschwindigkeit ist der größte Nachteil gegenüber den symmetrischen Verfahren.

Die Hybrid-Verschlüsselung ist eine Kombination beider Systeme, sie nutzt deren Vorteile und bietet somit eine interessante Alternative für die Datenübertragung.

Die Ver- bzw. Entschlüsselung der Daten geschieht mit einem symmetrischen Schlüssel, der aber zum Kommunikationspartner über einen sicheren Weg übertra-

gen werden muss. Dieser Schlüssel, auch „Session Key“ genannt, wird mit einem asymmetrischen Verfahren verschlüsselt, zur Gegenstelle übertragen und dort entschlüsselt. Danach kann er zur Ver- bzw. Entschlüsselung der Daten eingesetzt werden. D.h. die hybrid-Verfahren arbeiten mit symmetrischen Schlüssel, die sie vorab mit asymmetrischen Verfahren austauschen.

Beispiel: Personen $A$ und $B$ wollen eine hybrid-verschlüsselte Verbindung aufbauen

- $A$ generiert einen symmetrischen Schlüssel $k$
- $A$ verschlüsselt $k$ mit dem öffentlichen Schlüssel $p_b$ von $B$ $x = Enc_{p_b}(k)$
- $x$ wird zu $A$ evtl. über einen unsicheren Weg übertragen
- $B$ entschlüsselt $x$ mit dem privaten Schlüssel $s_b$ $k = Dec_{s_b}(x)$
- Die Daten $D$ können dann von $A$ und $B$ mit $k$ verschlüsselt $C = Enc_k(D)$, übertragen und wiederum entschlüsselt $D = Dec_k(C)$.

## 4.4 Hashfunktionen

Eine Hashfunktion $h$ ist eine Funktion, die Werte aus einem in der Regel sehr großen, im Prinzip unbeschränkten Urbildbereich in einen begrenzten Bildbereich abbildet. Daraus folgt bereits, dass die Funktion $h$ im allgemeinen nicht injektiv ist, d.h. es läßt sich nicht ausschließen, dass zwei Urbildwerte auf den selben Bildwert abgebildet werden
$h(k) = h(k')$ für $k \neq k'$
Man spricht dann von einer Kollision. Im einfachsten Fall kann man für eine Abbildung aus den ganzen Zahlen in die ganzen Zahlen und einen auf $m$ Werte beschränkten Bildbereich mit der mod-Funktion $h(k) = k \; mod \; m$ arbeiten, die Werte zwischen 0 und $m - 1$ liefert.

Eine Hashfunktion kann zur Gewährleistung der Datenintegrität eingesetzt werden. Die Grundidee ist, dass etwa im Fall der Datenübertragung der Sender aus der zu verschickenden Nachricht einen Hashwert berechnet und diesen mitüberträt. Der Empfänger berechnet mit der selben Funktion für die erhaltene Nachricht ebenfalls den Hashwert und vergleicht ihn mit dem empfangen. Aus der Gleichheit der Hashwerte, die wesentlich kürzer sind als die Nachrichten, kann man schnell auf eine unverfälschte Übertragung schließen, wenn die Wahrscheinlichkeit, dass eine beliebige Ursprungsnachricht und eine bewußt manipulierte oder durch technische Störeinflüsse zufällig veränderte Empfangsnachricht den selben Hashwert liefern, verschwindend gering ist.

Die Qualität einer Hashfunktion ergibt sich aus verschiedenen Kriterien.

**Kollisionsfreiheit bzw. Kollisionsarmut:** Der Bildbereich ist deutlich kleiner als der Urbildbereich, daher lassen sich Kollisionen nicht immer vermeiden. Die Hashfunktion soll aber kollisionsarm sein.

**Datenreduktion:** Der Hashwert sollte deutlich kleiner sein als die entsprechende Datenmenge.

**Streuung:** Ähnliche Datenwerte sollen völlig verschiedene Hashwerte ergeben. Damit wird es schwer, Kollisionen bewusst herbei zu führen.

**Surjektivität:** Die Hashfunktion soll den gesamten Bildbereich überstreichen und keine Werte vorab ausschließen.

**Effizienz:** Die Hashfunktion soll schnell und Resourcen schonend sein.

Einige bekannte in der Kryptographie eingesetzte Hash-Algorithmen sind MD4, MD5, SHA und RIPEMD-160, auf die hier nicht näher eingegangen wird.

## 4.5 Digitale Signatur

Moderne Angriffsmethoden, etwa die oben genannte Methode man-in-the-middle, können nicht nur dazu führen, dass übertragene Daten gezielt verfälscht werden, auch die angegebene Identität des Absenders kann eine andere sein als die des tatsächlichen Absenders, d.h. eine Person kann sich bewußt als jemand anders ausgeben. Analog zur handschriftlichen Unterschrift ist deshalb die digitale entstanden, die man meist als digitale Signatur bezeichnet.

Erst mehrere Jahre nach dem Entstehen des Konzeptes der digitalen Signatur kam es praktisch zum Einsatz. Die erste bekannte Methode, die die digitale Unterschrift ermöglicht, ist das „RSA signature scheme“, welches bis heute eine der praktischsten und vielseitigsten Verfahren geblieben ist. Für Deutschland ist das Signaturgesetz zu nennen, nicht zuletzt auch, weil einer der Väter dieses Gesetzes, Prof. Dr. Alexander Roßnagel, in Kassel lehrt [Roß96].

Mit der digitalen Signatur lässt sich auch prüfen, ob das Dokument sich nach der Unterschrift geändert hat oder nicht.

Die Abbildung 4.6 (Seite 37) zeigt, wie die digitale Signatur funktioniert. Bei der digitalen Signatur muss zwischen sechs Hauptbegriffen unterschieden werden:

**Digitale Signatur:** Verbindet den Inhalt mit dem Unterzeichner.

**Erzeugung der digitalen Signatur:** Ein Algorithmus generiert eine digitale Unterschrift für bestimmte Daten.

**Verifizierung der digitalen Signatur:** Ein Algorithmus prüft, ob die digitale Unterschrift authentisch ist.

**Das „digitale Signatur“-System:** Umfasst das Generieren und Verifizieren der digitalen Signatur.

**Unterzeichnungsprozess:** Umfasst das Generieren der Signatur und das Einbetten der Signatur in den Daten.

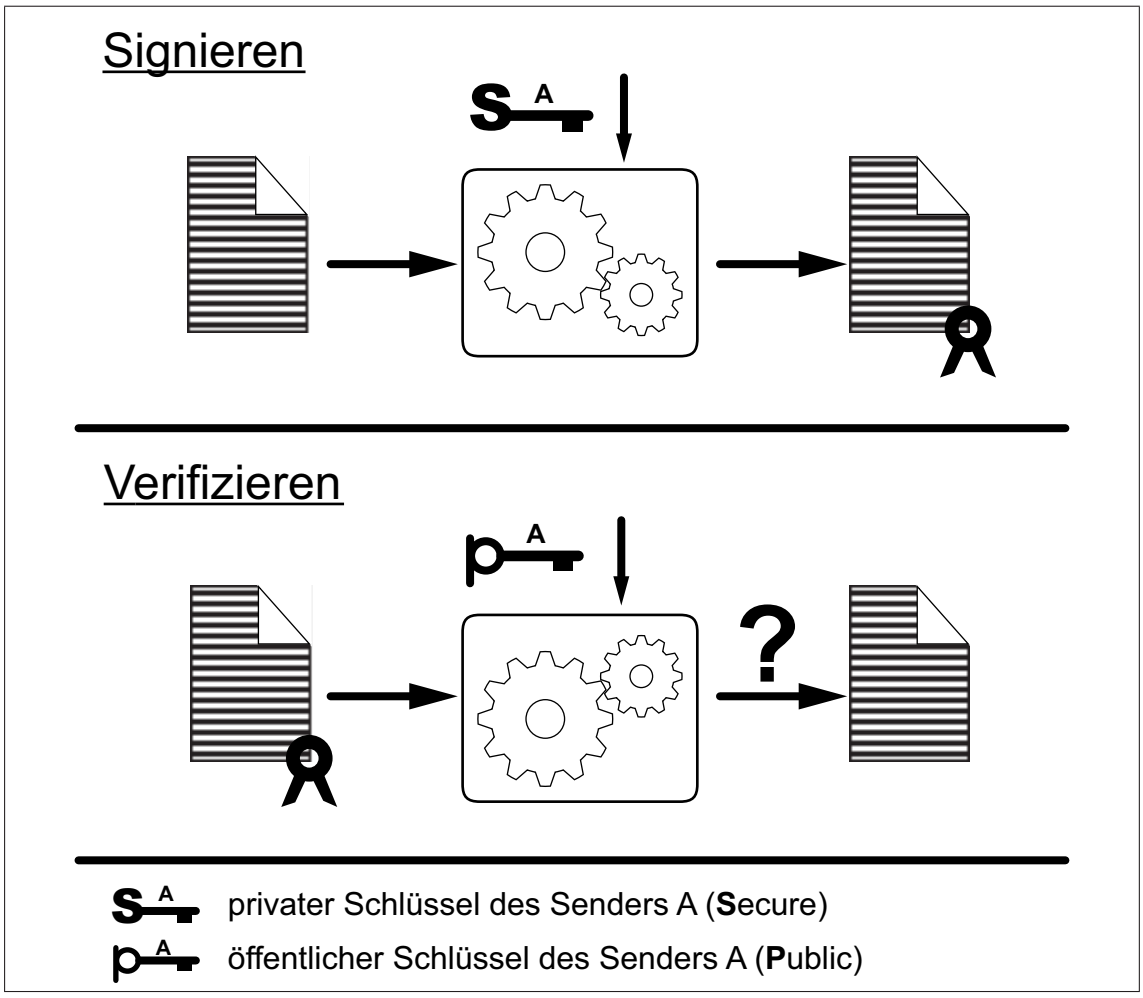



Abbildung 4.6: Signaturerzeugung und Signaturprüfung

**Verifizierungsprozess:** Umfasst das Prüfen der digitalen Signatur und das Wiederherstellen der ursprünglichen Daten aus den signierten.

Es sind zwei „digitale Signatur“-Systeme zu unterscheiden:

**System mit Anhang:** zum Verifizieren der Signatur sind die ursprünglichen Nutzdaten erforderlich. Bekannte Systeme mit Anhang sind unter anderen der Digital Signature Algorithm (DSA) und ElGamal.

**System mit Nachrichtwiederherstellung:** die Signatur kann geprüft werden ohne Kenntnis der ursprünglichen Nutzdaten. Diese können aus der Signatur wiederhergestellt werden. Als Beispiel dafür sind RSA und Rabin zu nennen.

# Kapitel 5

# Anonymität

Mit der Verbreitung des Internets und vor allem der elektronischen Geschäftsabwicklungen werden immer wieder unfreiwillig benutzerbezogene Spuren hinterlassen. Wäre man in der Lage, im Internet zu surfen, ohne seine Identität verraten zu müssen, könnte man das Problem umgehen. Die Geschäftspartner könnten aber dann benachteiligt sein, da ihnen unklar bleibt, ob die „anonyme“ Person auch der richtige Partner ist.

Die Lösung dieses Problems lautet „Sicherheit ohne Identifizierung“. Es sind viele sowohl praktische als auch theoretische Vorschläge entwickelt worden, wie man, ohne seine Identität bekanntgeben zu müssen, elektronische Vorgänge sicher abschließen kann.

Schon 1985 hat David Chaum in einem Artikel in den „Communications of the ACM“ [Cha85] auf die Wichtigkeit der Privatsphäre der Nutzer im Zuge der „Computerisierung“ hingewiesen.

> Computerization is robbing individuals of the ability to monitor and control the ways information about them is used. As organizations in both the private and the public sectors routinely exchange such information, individuals have no way of knowing if the information is inaccurate, obsolete, or otherwise inappropriate.

## 5.1 Begriffe

In [FP98] sind Unbeobachtbarkeit und Anonymität wie folgt definiert:

**Unbeobachtbarkeit** bedeutet, dass für einen Angreifer die Wahrscheinlichkeit des Auftretens eines Ereignisses (z.B. Senden einer Nachricht) *nach* jeder Beobachtung sowohl echt größer 0 als auch echt kleiner 1 ist. Perfekte Unbeobachtbarkeit bedeutet, dass für einen Angreifer die Wahrscheinlichkeit des Auftretens eines Ereignisses *vor und nach* jeder Beobachtung *gleich* ist.

**Anonymität** bedeutet, dass für einen Angreifer die Wahrscheinlichkeit, dass eine Instanz (z.B. eine Person) bei einem Ereignis eine bestimmte Rolle (z.B. Sender der Nachricht) wahrnimmt, *nach* jeder Beobachtung sowohl echt größer 0 als auch echt kleiner 1 ist. Perfekte Anonymität bedeutet, dass für einen Angreifer

die Wahrscheinlichkeit, dass eine Instanz bei einem Ereignis eine bestimmte Rolle wahrnimmt, *vor und nach* jeder Beobachtung *gleich* ist.

Weniger formal ausgedrückt bedeutet dies, dass die Belauschung bei gewährleisteter Unbeobachtbarkeit und Anonymität einem möglichen Angreifer keinen Informationsvorsprung liefert.

## 5.2 Standard-Techniken

Es wurden zahlreiche Techniken entwickelt, die eine anonyme Kommunikation ermöglichen. Hier werden die bekanntesten davon zusammengefasst ohne auf die Details einzugehen.

### 5.2.1 Schutz des Empfängers

Techniken, die für die Anonymität des Empfängers einer Nachricht sorgen:

**Verteilung (Broadcast):** Ist die Technik, die zum Beispiel im Rundfunkbereich eingesetzt wird. Die Nachrichten werden an alle Teilnehmer gesendet. Der Empfänger entscheidet, welche Nachricht und wann er sie empfangen will.

**Implizite Adressen:** Empfänger und Sender einigen sich über eine implizite Adresse, an der nur der Empfänger erkennen kann, dass die Nachricht für ihn ist. Für alle anderen ist die Adresse bedeutungslos. Eine implizite Adresse ist eine Bitkette, zum Beispiel die Zufallszahl bei einer Chiffreanzeige. Die impliziten Adressen werden eingesetzt, wenn eine Punkt-zu-Punkt-Kommunikation erfolgen soll.

### 5.2.2 Schutz des Senders

Techniken, die für die Anonymität des Senders einer Nachricht sorgen:

**Dummy Traffic:** Der Sender sendet permanent Daten. Wenn er keine echte Nachricht zu senden hat, werden leere Pakete geschickt, die für einen Dritten (evtl. Angreifer) nicht von richtigen Paketen zu unterscheiden sind. Somit lässt sich vermeiden, dass der Angreifer erfährt, ob der Sender eine Nachricht sendet oder gesendet hat.

**DC-Netz:** In einem DC-Netz („Dining Cryptographers"-Netz [Cha88]) kann ein Teilnehmer eine Nachricht senden, ohne als Sender erkannt zu werden. Der Teilnehmer ist innerhalb einer Gruppe anonym. Diese Gruppen werden Anonymitätsgruppen genannt.

### 5.2.3 Schutz der Kommunikationsbeziehung

Techniken, die die Kommunikationsbeziehung zwischen dem Sender und dem Empfänger verbergen:

**Mixe:** Der Prinzip der Mixe ist in [Cha81] genauer erläutert. Mixe werden in Vermittlungsnetzen verwendet. Sie sammeln empfangene Nachrichten, bis eine bestimmte Anzahl von Nachrichten von genügend vielen Absendern erreicht ist. Sie ignorieren die wiederholten Nachrichten, um Angriffe durch Nachrichtenwiederholungen zu verhindern. Danach werden die Nachrichten umkodiert, meistens wird asymmetrische Verschlüsselung angewendet. Zum Schluss werden sie in einer veränderten Reihenfolge in einem Schub an weitere Mixe ausgegeben(Abbildung 5.1, Seite 40).
Falls nicht genügend Nachrichten eingegangen sind, wird der Weiterversand nicht verzögert, sondern es werden künstliche Nachrichten (Dummy Traffic) erzeugt, um ein zeitnahes Verschicken der Nachrichten zu ermöglichen.

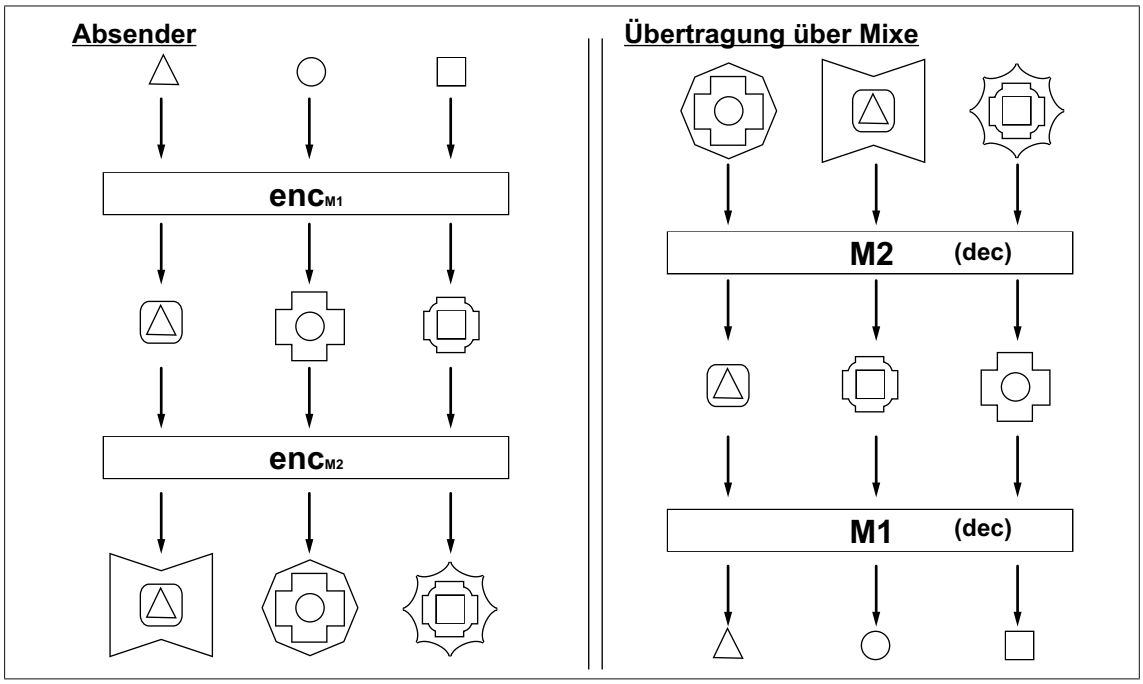



Abbildung 5.1: Das Mix-Konzept

### 5.2.4 Schutz der Aufenthaltsorte von mobilen Teilnehmern

Techniken, die verbergen, wo mobile Teilnehmer sich während der Datenübertragung befinden:

**Verteilung (Broadcast):** Nachrichten werden bewußt an mehr Zellen verschickt, als nötig, wenn das Vermittlungssystem die wahre Zelle kennt oder es wird an alle potentiellen Empfängerzellen gesendet und auf die Erkennung und Speicherung der Empfängerzelle verzichtet.

**Pseudonymisierung:** Die Aufenthaltsdaten werden unter einem Pseudonym gespeichert. Nur der Besitzer des Pseudonyms kennt das ihm zugeordnete Pseudonym.

**Mobilkommunikationsmixe:** Die Mixe schützen hier auch die Kommunikationsverbindung, hier wird jedoch die Verbindung zwischen der Aufenthaltsdaten-

bank und dem Aufenthaltsort des Teilnehmers geschützt. Ausführlichere Informationen dazu sind [FJP96] zu entnehmen.

## 5.3 Verfahren zur Anonymität im Web

Durch die starke Verbreitung des Internet und die zunehmende Abwicklung persönlicher Kommunikation darin gewinnen die Entwicklung und effiziente, anwendungsfreundliche Implementierung von Verfahren zur Unbeobachtbarkeit und Anonymität große Bedeutung. Hier werden die drei bekanntesten kurz erläutert.

### 5.3.1 Anonymizer

Technisch gesehen ist ein Anonymizer ein Proxy, der zwischen Benutzer und Zielrechner geschaltet wird. Da der Anonymizer anstelle des Benutzers mit dem Zielrechner kommuniziert, kann die Verbindung zum ursprünglichen Nutzer nicht ohne weiteres zurückverfolgt werden. Der Anonymizer unterscheidet sich jedoch vom herkömmlichen Web-Proxy, indem er alle potentiell personenbezogenen Daten aus dem Header der Anfragen entfernt.

Einer der größten Nachteile eines Anonymizer ist, dass er keine Verschlüsselung verwendet. Dadurch ist er gegen einen Angreifer, der die Kommunikation abhören kann bzw. Verkehrsanalysen durchführen kann, nicht sicher.

Alle Zugriffe und Daten werden von einer zentralen Instanz verwaltet, deswegen muss der Benutzer dem Betreiber des Anonymizer vollständig vertrauen, dass er die Verkehrs- und Interessendaten nicht sammelt oder weitergibt.

Zwischen dem Benutzer und dem Zielrechner können theoretisch mehrere Anonymizer hintereinander geschaltet werden. Nur der erste Anonymizer ist in der Lage, den ursprünglichen Sender der Abfragen zu identifizieren.

### 5.3.2 Crowds

Crowds ist eine Anwendungen auf der Basis des MIX-Modells und wird im [BSI] wie folgt beschrieben:

> Das Crowds-Verfahren wurde 1997 von Michael Reiter und Aviel Rubin bei AT&T entwickelt und in [RERU1997] beschrieben. Crowds wurde, anders als die vorhergehend besprochenen Verfahren, mit der Zielsetzung der Anonymisierung von Web-Zugriffen entwickelt und basiert nicht auf der Verwendung des MIX-Modells. Es soll aber trotzdem aufgrund seiner einfach anwendbaren Architektur und aufgrund seiner Bedeutung für die Praxis als Vergleich zu den Verfahren nach dem MIX-Modell hier vorgestellt werden.

**Funktionsweise** Eine Crowd oder Wolke kann man sich dabei als eine Anzahl von Internet-Nutzern vorstellen. Ein Nutzer innerhalb der Crowd wird durch einen Prozess auf seinem Rechner repräsentiert, den Reiter und Rubin jondo (sprich: John Doe, hier: Bezeichnung für einen unbekannten Teilnehmer) nennen. Der Nutzer startet jondo auf seinem Rechner. Nach dem Start kontaktiert der Prozess einen Server namens blender, um in die Crowd aufgenommen zu werden.

Nach Aufnahme in die Crowd wählt der Nutzer den jondo-Prozess als seinen Web-Proxy in den Einstellungen seines verwendeten Browsers. Dadurch werden alle Anfragen des Browsers zuerst von diesem Proxy-Prozess behandelt. Empfängt jondo die erste Anfrage des Browsers, initialisiert er die Auswahl eines Pfads zwischen dem die Informationen abrufenden Browser und dem Zielserver, von dem die Informationen abgerufen werden sollen. Genauer: Zuerst wird ein anderer in der Crowd verfügbarer jondo-Prozess zufällig ausgewählt, an den die Anfrage des Nutzers weiter geleitet wird. Wenn die Anfrage den ausgewählten jondo erreicht, entscheidet dieser, ob die Anfrage direkt an den Zielserver übergeben wird oder ob ein weiterer jondo-Prozess in der Crowd als Relais genutzt werden soll. Soll eine weitere Relais-Stelle durchlaufen werden wird das schon vom ersten jondo-Prozess durchgeführte zufällige Auswahlverfahren durch den ersten Relais-Prozess wiederholt und die Anfrage an den zweiten Relais-Prozess weiter geleitet. Dies wiederholt sich so oft, bis ein jondo-Prozess des Pfads die Entscheidung trifft, die Anfrage des Ursprungsprozesses an den Zielserver weiter zu leiten.

**Cypherpunk Remailer** und **MIXmaster** sind weitere auf das MIX-Modell basierende Anwendungen. Sie sind ebenfalls im [BSI] genauer beschrieben.

### 5.3.3 Onion Routing

Onion Routing umfasst auch weitere Bereiche neben dem Web-Bereich. Es ähnelt der Mixe-Lösung sehr stark.

Onion Routing arbeitet als Proxydienst. Auf der Seite des Senders ist ein „Initiator-Proxy“ geschaltet. Der „Responder-Proxy“ ist auf der Seite des Empfängers zu finden. Zwischen dem „Initiator-“ und dem „Responder-Proxy“ sind mehrere Onion-Router geschaltet, welche auch Nodes genannt werden.

Um eine Verzögerung zu vermeiden, wie die, die ein Mixe-System verursacht, wurden Änderungen an dem Grundkonzept durchgeführt, die zum Teil zu Lasten des erreichten Schutzes gingen.

Nach dem Festlegen der Route, d.h. über welche Nodes der Verbindungskanal aufgebaut wird, werden die zu sendenden Daten mit den öffentlichen Schlüsseln der Onion-Router mehrfach verschlüsselt, daher die Bezeichnung „Onion-Routing“ (im Deutschen „Zwiebel-Routing“). Man bezeichnet auch jede Verschlüsselung als „Schale“ (Abbildung 5.2, Seite 43).

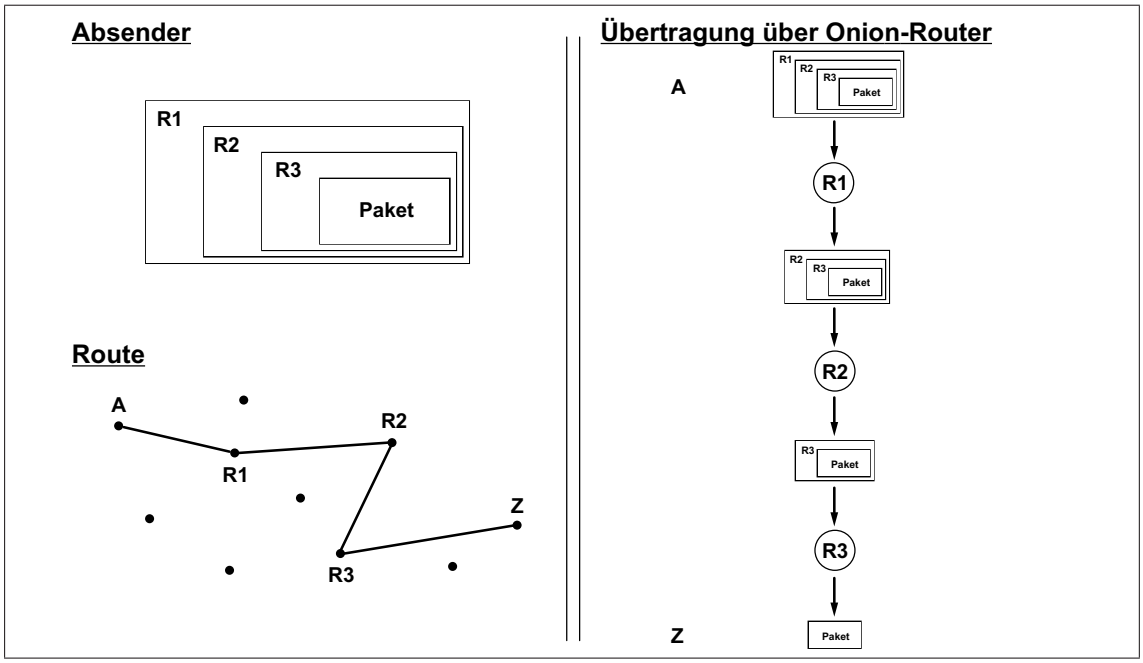



Abbildung 5.2: Das Onion-Router-Konzept

Jeder Onion-Router (z.B. R2 in der Abbildung 5.2, Seite 43) erfährt nur wer vor ihm in der Kette ist (R1) und an wen er senden muss (R3). Wer der richtige Absender ist, ist nur dem ersten Router bekannt. Ein einziger vertrauenswürdiger Onion-Router in der Kette verhindert die Feststellung der Kommunikationspartner.

Die Tabelle 5.1 (Seite 44) fasst die wesentlichen Unterschiede zwischen den oben genannten Verfahren zusammen.

Mittlerweile sind viele weitere Verfahren implementiert worden. Die oben beschriebene Anonymität ist meistens nur für die Datenübertragung im Netz nützlich. In den für diese Arbeit relevanten Fällen ist die Anonymität des Nutzers gegenüber dem Dienstanbieter meistens nicht gewünscht, denn Inhalte lassen sich vor dem Diensteanbieter verbergen und die Datenspur, die sich aus der Tatsache einer aufgebauten Verbindung ergibt (Abrechnungsdaten), ist für die geplante Anwendung unkritisch. Der Schutz der Identität des Nutzers während der Übertragung der Daten kann auf herkömmliche Art erfolgen und hat keinen Einfluss auf das in dieser Arbeit vorgestellte Verfahren.

| | **Zeitliche Verkettung** | **Verkettung über Inhalt** |
|---|---|---|
| **Anonymizer** | keine Vorkehrungen dagegen | keine Vorkehrungen, lediglich Headerinformationen werden entfernt |
| **Crowds** | keine Vorkehrungen, aber wenigstens Zusammenfassung von Anfragen in Jondos | keine Vorkehrungen, aber wenigstens sind Inhalte verschlüsselt |
| **Onion Routing** | schwache Vorkehrungen, lediglich Dummy Traffic zwischen Onion Routern | für Kanalaufbau keine Verkettung, für Datenaustausch jedoch Verkettung über Nachrichtenlänge möglich |

Tabelle 5.1: Vergleich der Verfahren

# Kapitel 6

# Problemstellung

Die modernen Authentifizierungsmethoden bieten ein hohes Maß an Sicherheit, leider sind aber auch zahlreiche Nachteile damit verbunden. In diesem Abschnitt wird, in Analogie zur Definition in der Rechtsprechung, zwischen Besitzer und Eigentümer unterschieden. Mit dem Eigentümer eines Objektes ist derjenige gemeint, dem das Objekt gehört. Der Besitzer eines Objektes dagegen ist derjenige, der tatsächlich über das Objekt verfügt.

Außerdem wird „Authentifizierungsmittel“ als allgemeiner Begriff für die Instanzen der Authentifizierungssystemen benutzt. Zum Beispiel die Gegenstände, die man zur Authentifizierung in einem „Besitz“-System benötigt, oder das Geheimnis bei einem „Wissen“-System.

## Handhabung

Der Verwaltungsaufwand ist einer der großen Nachteile der Authentifizierungssysteme. Damit ist die Verwaltung und das Aufbewahren der Schlüssel gemeint. Es muss sicher gestellt werden, dass die privaten Schlüssel geschützt aufbewahrt und dass die Benutzer-Schlüssel-Zuordnungen auch verlässlich geführt werden.

Die Mittel des „Wissen“-Systems (Kapitel 1), wie u.a. Passwort und PIN, sollten regelmäßig aktualisiert, gut organisiert und gut aufbewahrt werden, was das Merken für den normalen Verbraucher schwieriger und noch weitere Verfahren, wie eine Verschlüsselung, erforderlich macht.

Die Mittel des „Besitz“-Systems (Kapitel 1), wie u.a. Chipkarten und RFID, dürfen nicht in falsche Hände geraten, weil sie dem Besitzer den Zugang erlauben, auch wenn er nicht der Eigentümer ist. Es sei denn sie sind noch mit einem anderen Authentifizierungssystem bestückt, das die Identität des Besitzers genauer prüft.

Die Mittel des „Biometrie“-Systems (Kapitel 1), wie u.a. Gesichts- und Fingerabdruckerkennung, sind für den Verbraucher bei der Handhabung vorteilhafter als die oben genannten Systeme, aber meistens nicht flexibel. Sie sind personenbezogen und somit nicht übertragbar, was Vor- und Nachteile haben kann, und verlangen einen großen Aufwand bei gleichzeitig mangelnder Zuverlässlichkeit.

## Flexibilität

Bis auf einige Ausnahmen sind die Authentifizierungsmittel vorgangsspezifisch, das heißt ein Mittel kann nicht in allen Fällen eingesetzt werden, sondern nur für einen dafür vorgesehenen Zweck. So kann man in Deutschland mit einer Bahncard wenig andere Zahlungsvorgänge oder Identitätsnachweise führen, der Personalausweis war dagegen der in jedem Fall zu akzeptierende Nachweis einer Identität, hatte wegen seiner mangelnden Maschinenlesbarkeit aber im täglichen Geschäftsverkehr nur Nachteile. Die Folge davon ist, dass der Verbraucher im Besitz vieler Mitteln sein muss, obwohl sie sich meistens inhaltlich nur gering unterscheiden, was zu einem hohen Verwaltungsaufwand führt. Ideal wäre eine Art Identitätsmanagement, wie man es vom „single sign-on“ kennt (vgl. [RLS99, S. 181-191]), ohne in den Fehler zu verfallen, bei allen Anwendungen z.B. das selbe Passwort zu verwenden.

## Anonymität

Um die aktuellen Authentifizierungssysteme sicherer zu machen, werden immer mehr Benutzerdaten verlangt, was zur Aufgabe der Anonymität führt. Gleichzeitig wird eine Identifikation für immer mehr Anwendungen, sei es bargeldloser Einkauf, Telefonieren, Reisen, Online-Geschäfte, benötigt und es entstehen über Verknüpfungen der Datenerfassung ein Profil des Anwenders, das dieser in der Regel ausschließen will. Somit wäre es ideal, wenn der oben geforderte vereinfachte Identitätsnachweis nicht zu Lasten einer u.U. gewünschten Anonymität ginge.

## Lösungsvorschlag

Ein universelles System, das anonyme Authentifizierung ermöglicht und für den Verbraucher einfach zu bedienen ist, wäre die Lösung des Problems.
Das „**P**icture **b**ased **A**uthentication and **E**ncryption“ ***PbAE*** (Bildbasierte Authentifizierung und Verschlüsselung) System ist ein Vorschlag, wie so ein System aussehen könnte.

# Teil II

# Picture based Authentication and Encryption *PbAE*

# Kapitel 7

# Einführung in *PbAE*

## 7.1 Identitätsbasierte Kryptographie

Unter *Identitätsbasierter Kryptographie* versteht man ein Authentifizierungs- und Verschlüsselungssystem mittels asymmetrisches Schlüssel, bei dem der öffentliche Schlüssel eines Anwenders aus eindeutiger Information über die Identität des Anwenders besteht, z. B. aus seiner E-mail Adresse. Das erste Identitätsbasierte Kryptosystem war ein Signatursystem, das von Adi Shamir 1984 vorgeschlagen wurde [Sha85, Seiten 47-53] und das Anwendern erlaubte, digitale Signaturen allein auf der Basis öffentlicher Information, etwa einer bekannten Anwenderidentität, zu verifizieren. Moderne Schemata bedienen sich der Boneh/Franklin Weil-Paarbildung über elliptischen Kurven [BF01] und Cocks Verschlüsselungsschema [Coc01], das auf quadratischen Formen beruht (nach Wikipedia [idb]).

Der Gebrauch stellt sich wie folgt dar: ein Identitätsbasiertes System erlaubt es jedem, einen öffentlichen Schlüssel aus einer bekannten Identität, die als ASCII-Zeichenkette vorliegt, zu generieren. Ein vertrauenswürdiger Dritter (trusted third party), genannt *Private Key Generator* ($PKG$), generiert den zugehörigen privaten Schlüssel. Dazu veröffentlicht der PKG einen eigenen öffentlichen Generalschlüssel (public master key) und behält den geheimen Privatgeneralschlüssel (private master key) für sich.

Mit dem öffentlichen Generalschlüssel $TTP_{pub}$ kann nun jeder einen öffentlichen Schlüssel $C_{pub}$ für einen Kunden $C$ mit Identität $i$ errechnen, indem $i$ und $TTP_{pub}$ in die Berechnung als Parameter eingehen. Den zugehörigen privaten Schlüssel $C_{priv}$ des Kunden $C$ erhält dieser gegen Nachweis der Identität $i$ von dem $PKG$, der dafür seinen privaten Generalschlüssel $TTP_{priv}$ einsetzt. Wie bei asymmetrischen Schlüsselverfahren üblich, läßt sich der öffentliche Schlüssel eines Empfängers zur Verschlüsselung einer Nachricht verwenden, bzw. können Nachrichten mit dem privaten Schlüssel signiert werden. Der wesentliche Unterschied ist demnach, dass der öffentliche Schlüssel aus einem frei wählbaren ASCII-String, der zur Identität $i$ gehört, eindeutig festliegt. Die Bemerkung aus [BBDD05] stellt den Zusammenhang zwischen der Identität und dem öffentlichen Schlüssel genauer dar:

> ... the public key is calculated based on the identity. It is not the actual identity itself. This is a common misconception of identity based encryption. The advantage is that applications may perform this step behind

> the scenes so that the end user does not need to know the numerical value of the public key just the identity string.

Trotzdem kann der zugehörige private Schlüssel nach dem gegenwärtigen Stand der Erkenntnis nicht erraten werden. Für die Sicherheitsüberlegungen ist zu beachten, dass ein Angreifer sich vorab vom $PKG$ eine beliebig große Zahl - genauer polynomiell viele, vgl. [FS87], Remark 3 auf S. 189 - an öffentlichen-privaten Schlüsselpaaren, außer natürlich das anzugreifende Paar $C_{pub}$ und $C_{priv}$, generieren lassen kann, um sich damit eine Wissenbasis zu schaffen. Er kann dann (nach jetzigem Wissensstand allerdings vergeblich) versuchen, auf den privaten Generalschlüssel zu schließen [BF01]. Der Preis für die freie Wählbarkeit des öffentlichen Schlüssels ist die notwendige Mitwirkung einer vertrauenswürdigen dritten Seite, die im Übrigen auch immer eine Hintertür zur Entschlüsselung (key escrow) offenhält, da sie $C_{priv}$ jederzeit nochmals produzieren und an andere als den Kunden weitergeben könnte.

Der im Zusammenhang mit Identitätsbasierter Kryptographie immer wieder genannte Vorteil gegenüber den Verfahren mit asymmetrischen Schlüsseln ist der Wegfall der Schlüsselverteilung, genauer die Authentifizierung des öffentlichen Schlüssels. Dies bezieht sich auf die Gefahr, dass ein Gegner einem Absender einer geheimen Botschaft seinen eigenen öffentlichen Schlüssel als den des richtigen Empfängers vorspiegelt, die damit verschlüsselte Nachricht abfängt, mit seinem privaten Schlüssel entziffert und dann sogar die Nachricht mit dem öffentlichen Schlüssel des eigentlichen Empfängers codiert an diesen weitersendet, so dass die Tatsache des Abfangens und Entschlüsselns gar nicht auffällt.

Libert und Quisquater [LQ04] beschreiben den Stand der Technik wie folgt (dort zitierte Referenzen wurden ersetzt durch die Verweise in die Literaturliste dieser Arbeit):

> Identity based cryptography has become a very fashionable topic in the last couple of years. The motivation of this concept, introduced by Shamir in 1984 ([Sha85]), was to simplify key management and avoid the use of digital certificates. The trick was to let a public key be publicly and uniquely derivable from a human-memorizable binary sequence corresponding to an information non-ambiguously identifying its owner (e-mail address, IP address combined to a user name, social security number,...) while the associated private keys can only be computed by a trusted Private Key Generator (PKG) thanks to a master secret. This paradigm allows bypassing the trust problems that arise in traditional certificate-based public key infrastructures (PKIs). Indeed, since a public key 'is' its owner's identity, it becomes useless to bind them by a digital certificate. Although a PKG's public key still has to be certified, the need of digital certificates is really reduced as reasonably many users may depend on the same PKG.
> Since the concept's appearance in 1984, several practical identity based signature schemes (IBS) have been devised in the late 80's ([FS87], [QG88]) and also after 2001 ([CC03], [Hes02], [SOK00], [Pat02]). On the other hand, finding a practical identity based encryption scheme (IBE)

> remained an open challenge until 2001 when Boneh and Franklin ([BF01]) proposed to use bilinear maps over algebraic curves to elegantly solve the challenge. After that, these fashionable bilinear maps provided plenty of other applications (that are not listed here but their references can be found in [Bar05]) including various particular kinds of signatures: blind, ring, undeniable, proxy,etc.

Eine Bibliothek mit Algorithmen zur Implementierung eines Krytosystems auf der Basis bilinearer Paarbildungen findet sich unter http://crypto.stanford.edu/pbc/, zusätzlich auch ein Tutorial und Manual [Lyn06].

## 7.2 Verwendung eines Bilds als öffentlicher Schlüssel

In der Literatur zu Identitätsbasierter Verschlüsselung wird als Identifikator häufig die email-Adresse genannt, daneben die Sozialversicherungsnummer, Name/Vorname/Geburtsdatum. Boneh und Franklin [BF01] geben auch an, dass mit dem Identifikator ein Datum in die ASCII-Zeichenkette eingebaut werden kann, um die Gültigkeitsdauer des privaten Schlüssels zu begrenzen und dass sich über die ganzen Zahlen 1, 2, ..., $n$ als öffentlicher Schlüssel Sicherheitssysteme bauen lassen, bei denen sich Zuständigkeiten und Zugangsmöglichkeiten verteilen lassen.

Die Idee dieser Arbeit ist nun, für die frei wählbare ASCII-Zeichenkette das Bild des Kunden zu setzen, etwa als JPEG-Datei. Bei Anwendungen, bei denen die Identität der anwesenden Person bewiesen werden muss, könnte nun der Kunde $C$ sein Bild, das den öffentlichen Schlüssel $C_{pub}$ darstellt, einem Kontrolleur $G$ übergeben, der damit z.B. ein Ticket überprüft, das mit $C_{priv}$ signiert wurde. Die Zuordnung von $C_{pub}$ zur Person kann vom Kontrolleur per Augenschein erfolgen. Idealerweise muss $G$ für die Kontrolle nicht online sein.

Anders als in dem Zitat von Libert und Quisquater [LQ04] oben ist allerdings der öffentliche Schlüssel (das Bild etwa in Form einer JPG-Datei) nicht eine „human-memorizable binary sequence“. Genauso wenig ist ein Bild als öffentlicher Schlüssel „its owner's identity“, da es zu einem Besitzer viele ähnliche Bilder geben kann. Wir werden argumentieren, dass beide Eigenschaften den Gebrauch eines Bilds als öffentlichen Schlüssel in den genannten Szenarien nicht einschränkt.

## 7.3 Anwendungsszenarien

### 7.3.1 Personengebundene Transaktionen

Bei Transaktionen im Alltagsgeschäft gibt es Vorgänge, die strikt personengebunden (an die geprüfte oder beim Gebrauch überprüfbare Identität des Teilnehmers gebunden) sind. Dazu würde man die Verleihung von Zeugnissen, Beurkundungen, Vernehmungen und die Verkündigung von Urteile in Prozessen, Strafbefehle, die Auszahlung von Sozialleistungen, die Inanspruchnahme medizinischer Leistungen

mit Leistungserstattung aus Versicherungen (die nur für eine bestimmte Person leisten), eine Fahrerlaubnis, die Benutzung einer personengebundenen (nicht übertragbaren) Monats- oder Jahresfahrkarte, eine Einlaßkarte zu einer auf einen besonderen Personenkreis beschränkten Veranstaltung und andere Fälle zählen.

Andere Vorgänge erfordern keine Kenntnis des Teilnehmers solange das vereinbarte Entgeld für die gelieferte Leistung transferiert wurde. Darunter fallen die meisten Käufe, von dem Erwerb eines Brötchens, einer Kinokarte, einer Fahrkarte für die Straßenbahn, über den eines Fernsehers bis - prinzipiell - zum Autokauf. Gegebenenfalls sind Mindestaltersvorschriften zu erfüllen (etwa beim Bezug von Alkohol und Zigaretten, beim Eingehen von Kaufverträgen, die Geschäftsfähigkeit voraussetzen, bei der Benutzung eines Seniorentickets). In der Regel können diese Transaktionen auch ohne explizite Kenntnis der Identität des Teilnehmers abgewickelt werden. Auch Laufzeitverträge für Mobilfunk wären prinzipiell personenungebunden vorstellbar, wenn die Bezahlung (z.B. durch eine Firma für Ihre anonymisierten Mitarbeiter) gesichert wäre. Allerdings zeigt gerade das letzte Beispiel, dass solche Anonymisierungen, wie sie z. B. unregistrierte prepaid SIM-Karten darstellen, die Strafverfolgung behindern und daher vom Gesetzgeber verboten werden sollen.

Zwischen beiden Extremen gibt es eine Grauzone, in der heute häufig ein Personennachweis gefordert wird, obwohl ein solcher sachlich zunächst nicht gerechtfertigt erscheint oder die Notwendigkeit sich nur aus Wettbewerbs- oder Sicherheitsgründen (tatsächlichen und vorgeschobenen) ergibt. Dazu gehören personengebundene Flugtickets, personengebundene Eintrittkarten bei der letzten Fußball-WM, Garantieleistungen, die eine Registrierung voraussetzen, die Eröffnung eines Bankkontos, die Einrichtung eines Online-Zugangs bei manchen Betreibern.

In dieser Arbeit betrachten wir Szenarien der ersten Art, d. h. personengebundene Transaktionen, bei denen eine Kontrollperson die Identität prüfen muß, sei es zum Zeitpunkt der Leistungsvereinbarung oder vor und während der Leistungserbringung. Heute dient hierzu meist ein weitestgehend fälschungssicher gestalteter Pass oder Personalausweis mit Lichtbild. Obwohl dieser materielle (analoge) Beweis viele Vorteile hat, ist er eigentlich im Zeitalter allgegenwärtiger Digitalisierung und berührungslosem Datenaustausch ein Anachronismus.

Zugleich gibt es eine Tendenz, für die erste Art der Transaktion und die genannte Grauzone zusätzliche Identitätsnachweise zu schaffen, meist Karten mit Lichtbild, oft mit einer PIN für online-Geschäfte, die rollenbezogen sind. Beispiele sind Einkaufsausweise bestimmter Großmärkte (Rolle Käufer), Bahncard-Ausweise (Rolle Bahnfahrer mit bezahltem Rabattanspruch), Fitnessstudios (Rolle Studiobesucher), Gesundheitskarte, ggf. Sozialversicherungsausweis, Mitarbeiterausweis.

Hier soll nun auf der Grundlage der oben genannten Identitätsbasierten Kryptographie in der dort vorgeschlagenen Variante mit einem Bild als öffentlichem Schlüssel experimentell versucht werden, ein Mobilfunkgerät (Handy), einen PDA oder ein vergleichbares Gerät an die Stelle dieser rollenbasierten, mit einem Bild verbundene Identitätsnachweise per Ausweis treten zu lassen. Dabei könnten klassische Anwendungen, bei denen neben dem Bild auch andere Identifikatoren wie Name, Anschrift,

Sozialversicherungsnummer, usw. bekannt sind, in Frage kommen, als auch Anwendungen, bei denen ein Konzept eines bildbasierten Authentifikators ohne direkte Identitätsangabe zum Einsatz kommt. Letzteres entspräche einem fälschungssicheren Bildausweis ohne weitere Angabe des Eigentümers. Das klingt zunächst paradox, weil man es gewohnt ist, zu jedem Gesicht einen Namen, oft auch eine Anschrift, die Kundennummer usw. genannt zu bekommen.

Tatsächlich genügt aber bei den gegenwärtigen Überlegungen ein einziger weiterer Identifikator auf der Karte (oder in unserer Entwicklung, dem Gerät) um anwendungsspezifisch und unter Kontrolle des Besitzers Verknüpfungen zu der behaupteten Rollenidentität (ohne deren explizite Benennung) herzustellen. Im Prinzip haben wir es also mit einer biometrischen Gesichtserkennung zu tun, nur dass eine kontrollierende Person (ein Mensch also) diese Erkennung durch Vergleich eines gespeicherten Bildes mit einer anwesenden Person leistet. Dies ist momentan deutlich sicherer angesichts der noch recht hohen Fehlerraten der Bilderkennung ad hoc aufgenommener Personen.

### 7.3.2 Anwendung nichtübertragbare Monatskarte

Wir betrachten zunächst eine klassische Anwendung bei der ein Kunde $C$ sich bei einem Dienstleister $D$ mit seinem Bild $B$, das als öffentlicher Schlüssel $C_{pub}$ dient, registrieren läßt. Dienstleister $D$ speichert dazu auch Bild $B$. Natürlich ist $B$ genau das Bild, das Kunde $C$ bei der vertrauenswürdigen dritten Seite ($PKG$) verwendet hat, um sich $C_{priv}$ generieren zu lassen, den privaten Schlüssel, den der Kunde nicht aus der Hand gibt.

Kunde $C$ bestellt nun bei $D$ eine Dienstleistung, die wir Ticket $T$ nennen wollen, z.B. eine Monatskarte für die Straßenbahn. Die Bestellung kann online erfolgen. Hat $C$ bei $D$ sein Bild hinterlegt, können sich $C$ und $D$ über ein Challenge-Response Verfahren wie gewohnt identifizieren indem $D$ an $C$ eine Zufallszahl schickt, die $C$ mit dem privaten Schlüssel signiert und $D$ mit dem Bild prüft. Allerdings wird hier nur der Besitz des privaten Schlüssels geprüft, die Übereinstimmung mit der Person erfolgt nicht. Da das Ticket aber für Kunden $C$ ausgestellt wird und nur von ihm nutzbar ist, ist dies keine Einschränkung für das Verfahren.

Dienstleister $D$ schickt nun an Kunden $C$ das Ticket $T$ im Klartext. Der Kunde prüft die darauf gemachten Angaben und signiert es dann mit seinem privaten Schlüssel. Das signierte Ticket schickt er an den Dienstleister zurück.

Der Dienstleister prüft nun die Signatur mit dem hinterlegten Bild und überzeugt sich gleichzeitig, dass das Ticket nicht verfälscht wurde. Danach signiert der Dienstleister das bereits vom Kunden signierte Ticket mit dem privaten Schlüssel des Dienstleisters und schickt dieses zweifach signierte Ticket zurück an den Kunden, der es auf seinem portablen Gerät, z.B. Handy, speichert. Zuvor kann sich der Kunde mit dem öffentlichen Schlüssel des Dienstleisters davon überzeugen, dass das zurückgeschickte Ticket tatsächlich vom Dienstleister kommt und unverfälscht ist.

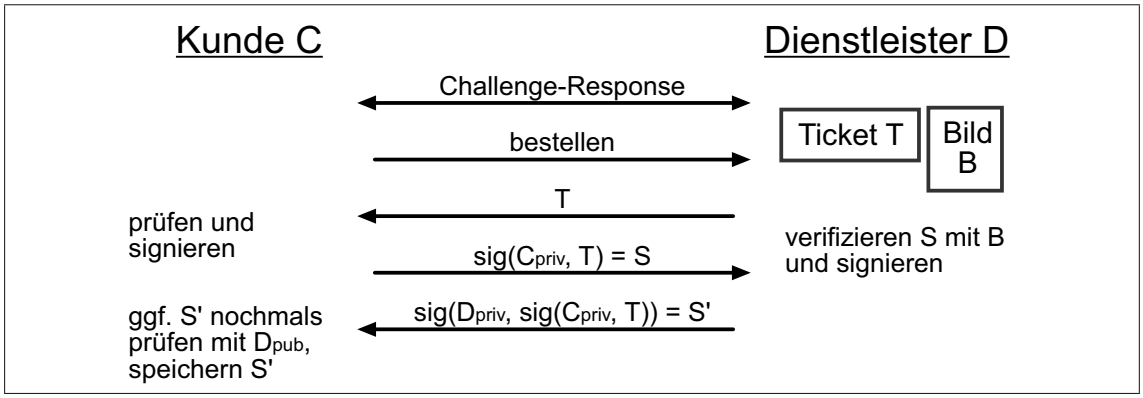



Abbildung 7.1: Ticketausstellung

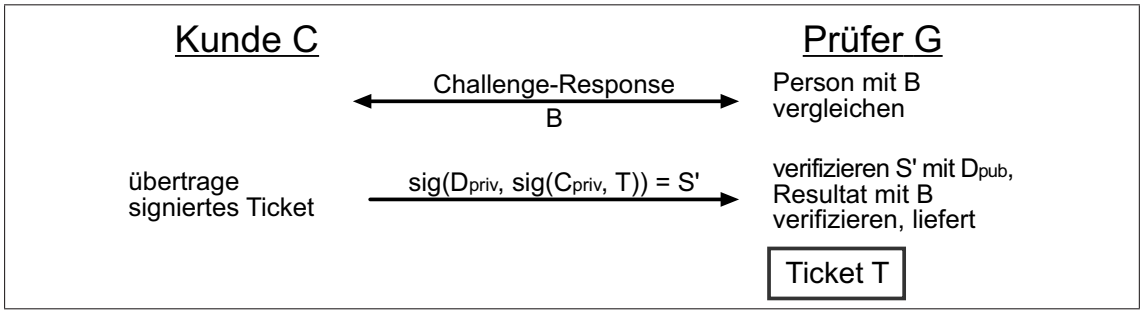



Abbildung 7.2: Ticketprüfung

Bezeichne $sig(k, t) = s$ das Signieren eines Textes $t$ mittels Schlüssel $k$ zur Signatur $s$ und $ver(s)$ die Verifikation der Signatur $s$. Dann zeigt die Abbildung 7.1 (Seite 53) den Vorgang der Ticketausstellung.

Der Prüfvorgang geschieht dann in offensichtlicher Weise, wobei der Prüfer (guard) $G$ nicht online sein muss. Der Prüfer weist sich per Challenge-Response (auch ID-basiert, aber nicht notwendigerweise mit einem Bild als öffentlichem Schlüssel) als Mitarbeiter des Dienstleisters $D$ aus. Hierzu könnte Kunde $C$ dem Prüfer $G$ eine mit einem öffentlichen Tagesschlüssel des Dienstleisters verschlüsselte Zufallszahl schicken (ggf. zu hause vorberechnet und auf dem Handy in der verschlüsselten Version und im Klartext gespeichert, die der Prüfer mit dem privaten Tagesschlüssel des Dienstleisters entschlüsselt und zurückschickt (Abbildung 7.2, Seite 53).

Wie man sieht, erfolgt die Prüfung ausschließlich auf dem Gerät des Prüfers, etwa einem Tablett-PC, der bezüglich Rechenleistung in etwa einem modernen PC entspricht und deutlich über der Leistung eines Mobiltelefons liegt. Auch dürften hier Bildschirmauflösungen für das Bild möglich sein, die einen Abgleich mit dem Kunden auch unter ungünstigen Lichtverhältnissen ermöglichen, was auf dem Display eines Handys nicht gegeben wäre.

Nachteil dieser Lösung ist, dass der Prüfer $G$ prinzipiell das Bild $B$ auf seinem Gerät speichern kann. Wir argumentieren, dass angesichts der weiten Verbreitung von offenen und versteckten Kameras, jeder sich auch auf diesem Wege ein Bild des Kunden verschaffen könnte.

Grundsätzlich ist diese Lösung nicht sehr verschieden von einer Lösung mit einem vom Dienstleister signierten digitalen Ticket, das einen Identitätshinweis enthält, etwa den Kundennamen. Der Prüfer muss dann über einen Ausweis mit Lichtbild und Namen konventionell die personengebundene Berechtigung prüfen. Auch die Sicherheit der Lösung ist praktisch identisch. Ein gestohlenes Gerät samt Zugriff auf den privaten Schlüssel erlaubt dem Dieb keine Fälschung der Tickets zu seinen Gunsten, solange diese mit dem privaten Schlüssel des Dienstleisters signiert sind. Diese Aussagen gelten natürlich immer unter der Annahme, dass nach gegenwärtigem Stand der Kryptographie kein Angreifer ein ihm ähnlichsehendes Bild $B'$ produzieren kann, das sich in der Verwendung als Signatur wie ein anderes Bild $B$ verhält, d.h. als öffentlicher Schlüssel $B' = C_{pub}$ zu einem privaten Schlüssel $C_{priv}$ paßt, der aber vom PKG für ein Bild $B \neq B'$ produziert wurde. Zweitens wird Bijektivität vorausgesetzt, sodass es grundsätzlich zu einem privaten Schlüssel genau einen öffentlichen Schlüssel gibt, d. h. die Bilder werden nicht vorab mittels Hash auf einen kleineren Bildraum abgebildet.

### 7.3.3 Authentifizierung ohne Identitätspreisgabe

Wie gezeigt, erfährt der Prüfer nichts über die Identität des Kunden, wenn dessen Identität z. B. nicht als Name oder ähnliches auf dem Ticket vermerkt ist. Das wäre z. B. bei einer nichtübertragbaren Monatskarte durchaus möglich, wenn der Nutzer sich ausschließlich über das Bild, das zur Signatur verwendet wurde, authentifiziert. Selbst bei der Ausstellung des Tickets könnte der Zugriff zunächst über das eingesandte Bild erfolgen, d.h. der Zugriff auf die Kundendaten würde über ein Bild als Schlüssel erfolgen.

Da mit der Bestellung einer Dienstleistung in der Regel ein Bezahlvorgang verbunden ist, der wiederum zur Identifikation herangezogen würde, müßte die Abbuchung über einen Dritten erfolgen, dem das Bild als Schlüssel angeboten wird und der den Transfer des geschuldeten Geldbetrags veranlaßt.

Interessant, aber hier nicht weiters untersucht, wäre die Möglichkeit, nur einen Teil des Bildes des Kunden als öffentlichen Schlüssel zu verwenden und diesen Teil erst zum Zeitpunkt der visuellen Prüfung zu einem ganzen brauchbaren Bild zusammenzusetzen. Die Teilung könnte ganz einfach darin bestehen, nur jedes zweite Byte des Bildes zu verwenden, oder es könnten aufwendigere Überlagerungstechniken zum Einsatz kommen, wie etwa die von Andreas Klein [Kle07] vorgeschlagenen. Damit nicht doch der Prüfer ein vollständiges speicherbares Bild erhält, müßte die Anzeige auf dem Gerät des Kunden erfolgen oder per Projektion zusammengesetzt werden. Alle diese Lösungen sind aber unbefriedigend in der praktischen Umsetzung und werden deshalb hier nicht weiter verfolgt. Genauso wäre der Einsatz von Methoden des Digital Rights Managements (DRM) vorstellbar, die eine Anzeige nach Ablauf eines Verfallsdatums oder ohne spezielle Freischaltung verhindern. Diese kommen heute im Bereich digitale Medien (Pay TV) zum Einsatz, lassen sich aber meist mit genügend technischem Aufwand umgehen und werden hier nicht betrachtet.

### 7.3.4 Verwendung des PKG im Zusammenhang mit rollenbasierter Authentifikation

Wie beschrieben, kann ein Kunde C sich über ein Handy oder PDA per Bild authentifizieren. Dies kann innerhalb vielfältiger Rollenidentitäten geschehen, etwa als Bürger, als Reisender, oder als Käufer. Dabei muß er nicht notwendigerweise andere Identifikatoren (seinen Namen oder seine Kreditkartennummer) preisgeben. Da er nicht sicherstellen kann, am Ort der Kontrolle Netzzugang zu haben und da dies u. U. ihn Kommunikationsgebühren kostet, sollte die Nutzung offline möglich sein. Bei vorbestellten Dienstleistungen ist dies mit den signierten Tickets möglich wie beschrieben.

Sein Bild betrachtet der Kunde als nicht geheim. Wie bereits erwähnt, muß er sein Gesicht in den oben genannten Szenarien zeigen und muß auch damit rechnen, dass eine Kameraüberwachung sein Gesicht erfaßt. Andererseits hat er kein Interesse daran, dass der Kontrolleur massenhaft und leicht Bilder der kontrollierten Personen speichert, da sich daraus Profile ableiten lassen (wer Freitags immer den Zug 15:37 Uhr von Kassel nach Frankfurt nimmt, wer auf dem Rock-gegen-Rechts Konzert war, usw.). Allerdings besteht diese Möglichkeit schon heute durch verdeckte Beobachtung. Eine Verschlechterung gegenüber dem jetzigen Zustand sollte verhindert werden. Denkbar wäre der Einsatz spezieller Kontrollgeräte mit stark begrenzten Speichermöglichkeiten oder solchen, bei denen der Datentransfer aus diesen Geräten heraus über Signaturen kontrolliert wird.

Schwieriger ist schon die Frage, ob ein Kunde die Speicherung seines Bildes auf den Servern vieler Dienstleister genehmigen will. So speichert die Deutsche Bahn das Porträt der Kunden zur leichteren Generierung einer neuen Bahncard, ein Fitnessstudio, das sich beim Datenschutz unter Umständen schwerer tut, könnte dies auch machen, ein Großmarktunternehmen könnte darauf verzichten und mit jedem neuen Jahresausweis ein neues Bild aufnehmen. Im letzten Fall brächte eine Speicherung des Bildes eine Verschlechterung im Schutz der Privatsphäre.

Bei ordentlicher Trennung der gespeicherten Bilder von sonstiger Information unter Beachtung datenschutzrechtlicher Auflagen erscheinen die Mißbrauchsmöglichkeiten gering. Über Zugangskontrollen läßt sich erreichen, dass Unbefugte nicht Scans über Bildbestände vornehmen. In jedem Fall wäre die explizite Verwendung des eigenen Bildes ohne weitere Identitätsangaben eine Verbesserung gegenüber der jetzigen Situation, in der eine geprüfte Identität mit heimlich erfassten Bildern verknüpft werden kann.

## 7.4 Besonderheiten einer bildbasierten Kryptographie

In der Einleitung zu diesem Verfahren wurde erwähnt, dass man unter *Identitätsbasierter Kryptographie* ein Authentifizierungssystem mittels asymmetrischer Schlüssel versteht, bei dem der öffentliche Schlüssel eines Anwenders aus eindeutiger Information über die Identität des Anwenders besteht, z. B. der E-mail Adresse oder z. B.

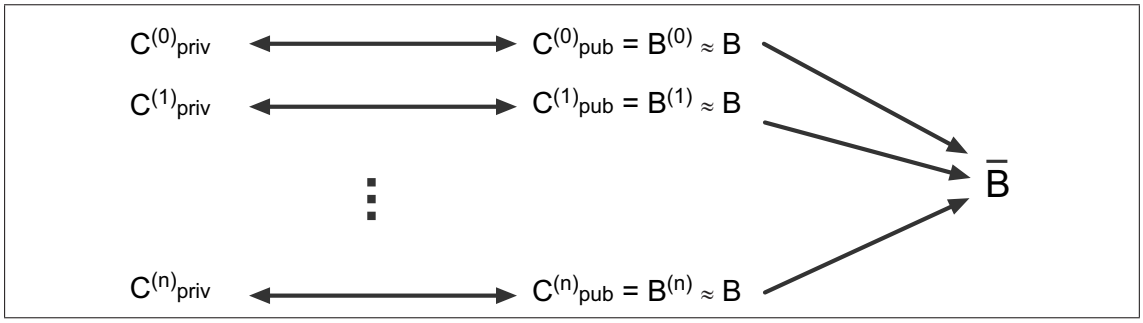



Abbildung 7.3: Bildmenge

einer Sozialversicherungsnummer. Setzt man an deren Stelle nun ein Bild, muss man berücksichtigen, dass man das Bild an einigen (sogar relativ vielen) Stellen mehr oder weniger stark verändern kann, ohne dass dies visuell auffällig wäre. Man denke dabei an die Manipulation einiger Pixel oder eine generelle Änderung im Farbraum des Bildes, beides ist oft mit bloßem Auge nicht zu erkennen.

Wegen der kryptographischen Eigenschaften wird ein so manipuliertes Bild, das als öffentlicher Schlüssel dient, einen anderen privaten Schlüssel generieren, der nicht zum ursprünglihen Bild (öffentlichem Schlüssel) paßt. Die Frage lautet, ob sich durch diese Besonderheit Gefahren ergeben.

Sei Bild $B = B^{(0)}$ das ursprüngliche (unverfälschte) Bild und sei Bild $B^{(i)}$ das i'te verfälschte Bild, das man in der Regel visuell nicht von $B$ unterscheiden kann, $1 \leq i \leq n$. Zusammen bilden sie eine Bildmenge (Abbildung 7.3 Seite 56), deren Repräsentant $\bar{B}$ sei. Wie groß diese Menge ist, läßt sich nicht bestimmen, sondern hängt davon ab, unter welchen Bedingungen ein Individuum ein Bild betrachtet. Eine genaue Grenze ist für die folgenden Überlegungen auch irrelevant, wie sich herausstellt.

Festzuhalten ist, dass ein Anwender **beim Verschlüsseln** mit einem angebotenen Bild $B^{(i)}$ des Empfängers diesen öffentlichen Schlüssel nicht als Fälschung zu einem ihm bekannten Bild $B$ erkennen kann. Dabei kann derjenige, der dieses gefälschte Bild dem Absender einer geheimen Botschaft untergeschoben hat, daraus nur Nutzen ziehen, wenn er den privaten Schlüssel $C^{(i)}_{priv}$ besitzt. Diesen müßte ihm der PKG erzeugt haben. Somit kommt auf dieses die zusätzliche Aufgabe zu, keine ähnlichen Schlüsselpaare zu generieren. Diese Aufgabe ist bei Bildern deutlich aufwendiger als zu verhindern, dass für eine geschickt verfälschte email-Adresse ein privater Schlüssels generiert und verschickt wird, obwohl für die unverfäschte email-Adresse bereits ein privater Schlüssel generiert wurde. Die Mißbrauchsmöglichkeit ist aber wieder auf man-in-the-middle Attacken beschränkt, bei denen dem Absender einer geheimen Botschaft ein gefälschter öffentlicher Schlüssel untergeschoben wird. Eine IBE unter Verwendung eines Bildes stellt damit keine Verschlechterung dar.

Bei der **Verwendung als Signatur** kann ein Prüfer $G$ nicht visuell entscheiden, ob das gezeigte Bild tatsächlich der öffentliche Schlüssel der Person vor ihm ist. Fest steht bei erfolgreicher Prüfung nur, dass das gezeigte Bild $B^{(j)} = C^{(j)}_{pub}$ zur

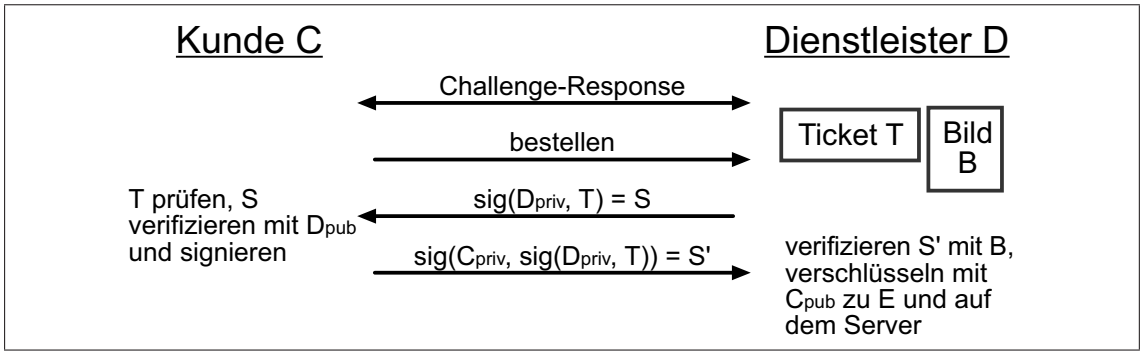



Abbildung 7.4: „Verschlüsseltes Ticket“-ausstellung

Signatur und damit zu einem $C_{priv}^{(j)}$ paßt, mit dem ein Dokument signiert wurde. Wichtig ist allerdings, dass bei einem manipulierten Bild $B^{(j)}$, das visuell von $B$ nicht unterscheidbar ist, immer noch die richtige Person vor dem Prüfer $G$ stehen müßte. Da der Kunde seinen privaten Schlüssel nicht aus der Hand gibt, müßte der Dienstleister mutwillig ein verfälschtes Schlüsselpaar zur Signatur einsetzen. Eine Motivation hierfür ist nicht erkennbar und würde auch wieder voraussetzen, dass der PKG dem manipulationswilligen Dienstleister $D$ einen privaten Schlüssel $C_{priv}^{(j)}$ zu einem $C_{pub}^{(j)} = B^{(j)}$ generiert, obwohl dem Vertrauenszentrum zuvor bereits ein visuell nicht unterscheidbarer Schlüssel $C_{pub}^{(0)} = B$ vorlag.

Generell wäre zu überlegen, ob der PKG nicht sogar aus einer Serie von Bildern mehrere Paare aus privaten und öffentlichen Schlüsseln erzeugen sollte, mit denen die Person dann ein eigenes Identitätsmanagement betreiben könnte. Zu prüfen wäre auch, ob der PKG in jedem Fall die Identität der Person prüft, die ein Bild vorlegt und wem gegenüber sie diese Identität auf Verlangen preisgibt.

## 7.5 Verwendung bildbasierter Kryptographie für die Verschlüsselung

In den oben genannten Szenarien werden die Tickets dem Kunden ausgehändigt worden, der sie zur Prüfung zur Verfügung stellen muss. Wäre das nicht möglich, bzw. nicht erwünscht, und würden die Tickets auf einem Server gespeichert, kann man sie schützen, indem man sie vor dem Speichern verschlüsseln lässt. Diese Funktion lässt sich ganz einfach mit dem hier vorgeschlagenen System realisieren. Mit dem öffentlichen Schlüssel des Kunden $C_{pub}$ (das Bild $B$) werden die Tickets von dem Dienstleister $D$ verschlüsselt und auf dem Server gespeichert. Der Prüfer $G$ kann sie bei Bedarf aus dem Server holen und sie vom Kunden entschlüsseln lassen, dann kann die Prüfung wie oben beschrieben stattfinden.

Bezeichne $enc(k, t) = E$ das Verschlüsseln eines Textes $t$ mittels Schlüssel $k$ zum chiffrierten Text $E$ und $dec(k, c)$ die Entschlüsselung des chiffrierten Textes $E$ mittels Schlüssel $k$. Dann zeigt die Abbildung 7.4 (Seite 57) den Vorgang solcher Ticketausstellung. Der Verlauf der Ticketprüfung kann der Abbildung 7.5 (Seite 58) entnommen werden.

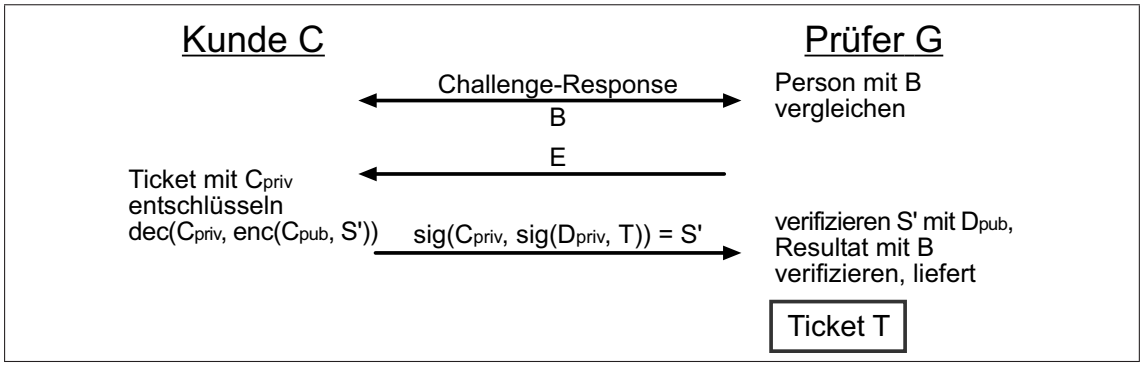



Abbildung 7.5: „Verschlüsseltes Ticket“-prüfung

Der Nachteil dieser Lösung ist, dass der Kontrolleur online sein muss, um das verschlüsselte Ticket aus dem Server abzurufen.

# Kapitel 8

# Implementierungsaspekte und Laufzeitverhalten

Die *Identitätsbasierte Kryptographie* ist ein asymmetrisches kryptographisches System, bei dem der öffentliche Schlüssel keine zufällige Zahl ist, sondern eine eindeutige Information über die Identität des Anwenders. Anstatt eine E-mail als Schlüssel oder Teil des Schlüssels zu verwenden, wie es in den meisten Szenarien der Fall ist, besteht der Identität in dem ***PbAE***-System aus einem Bild des Anwenders bzw. den Daten, die ein Gesichtserkennungssystem über den Anwender liefert.

In diesem Kapitel wird folgendes untersucht:

- Ist der Einsatz solcher Daten als öffentlicher Schlüssel technisch machbar?
- Welchen Einfluss hat die Schlüssellänge auf das System?
- Wie sicher ist der Bild-basierte Schlüssel?

## 8.1 Einleitung

Um die Plattform-Unabhängigkeit des Systems zu erreichen, ist das System in Java implementiert, was aber keine Voraussetzung ist, sondern nur ein Vorschlag. Jede Programmiersprache, die evtl. mit Hilfe Zusatzbibliotheken die Kommunikationsschnittstellen unterstützt, die in dem System zum Einsatz kommen, wie Sockets bei den Internet/Intranet-Verbindungen oder Datagramm bei der Bluetooth-Verbindung, kann eingesetzt werden.

J2SE bietet keine Klassen für die identitätsbasierte Kryptographie, deswegen war eine Zusatzbibliothek notwendig.

Die Arbeitsgruppe „Computer Security and Cryptography Group“ der Abteilung „Department of Computer Science“ der „National University of Ireland“ in Maynooth hat die Bibliothek „Identity Based Encryption JCA Provider“ implementiert, welche die identitätsbasierte Kryptographie unter Java unterstützt.

## 8.2 Einsatz des IBE-JCA Providers

Die Bibliothek lässt sich einfach in eigenen Projekten einbinden und stellt zahlreiche Funktionen zur Schlüssel-Generierung, Ver-, Entschlüsselung und Signatur von Daten zur Verfügung.

```
import java.math;
import java.security;
import javax.crypto;
import nuim.cs.crypto.ibe;
import nuim.cs.crypto.bilinear;
import nuim.cs.crypto.blitz.point;
```

Quellcode 8.1: IBE-JCA Provider einbinden

Als erstes muss das System initialisiert werden. Dabei wird ein $IBE - provider$ erzeugt. Eine Instanz des Klasse *Cipher* ermöglicht, je nach Einstellung (s.u.), das Ver- bzw. Entschlüsseln der Daten. Für das Generieren des Schlüsselpaars, bestehend aus einem privaten und einem öffentlichen Schlüssel, ist ein *KeyPairGenerator*-objekt nötig.

```
private void init(){
  provider = new IbeProvider();
  // get the ibe cipher
  cipher = null;
  try {
    // Cipher-instance for IBE
    cipher=Cipher.getInstance(IbeProvider.IBE, provider);
  }
  catch( NoSuchAlgorithmException nsae ) {
    // handle exception
    nsae.printStackTrace();
  }
  catch( NoSuchPaddingException nspe ) {
    // handle exception
    nspe.printStackTrace();
  }

  // get the ibe key pair generator
  kpg = null;
  try {
    kpg=KeyPairGenerator.getInstance(IbeProvider.IBE,provider);
  }
  catch( NoSuchAlgorithmException nsae ) {
    // handle exception
    nsae.printStackTrace();
  }
}
```

Quellcode 8.2: IBE-System Initialisieren

Danach werden weitere Systemkomponenten generiert, wie einen zufälligen Hashwert *hash* und eine *map*, die eine elliptische Kurve repräsentiert.

```
private void initSystem(){
  // initialise the message digest to be used
  hash = null;
  try {
      hash = MessageDigest.getInstance( "MD5" );
    }
    catch( NoSuchAlgorithmException nsae ) {
      // handle exception
      nsae.printStackTrace();
    }

  // set the system parameters
  map = new ModifiedTatePairing();

  boolean fieldTooSmall = false;
  do {
    fieldTooSmall = false;
    // get a count of the maximum possible number of
    // points that we could use, i.e. p / kappa where
    // p is the upper limit of finite field Fp
    BigInteger maxPts =
        map.getCurve().getField().getChar().
        divide( map.getCurve().getKappa() );
    int bytes = maxPts.bitLength() / 8;
    if( bytes < hash.getDigestLength() ) {
        map = new ModifiedTatePairing();
        fieldTooSmall = true;
    }
  }
  while( fieldTooSmall );
}
```

Quellcode 8.3: System-Variablen definieren

Anhand der Kurve und einer zufälligen Zahl *newSecureRandom*() wird der Master-Schlüssel erzeugt. Die *map*, *hash* und der *masterKey* sind die Parameter, die das System eindeutig machen und es von den anderen unterscheiden lassen. Bei mehrmaliger und sicherer Nutzung des Systems müssen diese Parameter gut aufbewahrt werden.

```
private void createParams(){
  // create the parameters
  masterKey =
    new BigInteger(map.getQ().bitLength()-1,new SecureRandom());

  systemParameters =
    new IbeSystemParameters(map, hash, masterKey);
}
```

Quellcode 8.4: Systemparameter einstellen

Sind die Systemparameter erfolgreich initialisiert, kann das System zum Einsatz kommen. Aus einer „Identität“, die aus einer Zeichenkette besteht, werden Schlüssel-

Parameter generiert, die wiederum das Generieren des zur „Identität“ passenden öffentlichen Schlüssels ermöglicht.

```
private void loadPublicKey(){
  // set the identity as a string
  //identity = new String( "mustermann@musterserver.de" );
  // or reading it from a file
  identity = new String(readFile("keyfile"));

  keyParameters = new IbeKeyParameters( hash, identity );
  // get the public key based on the identity
  publicKey = new IbePublicKey(keyParameters.getPublicKey());
}
```

Quellcode 8.5: Öffentlicher Schlüssel laden

Zum Verschlüsseln wird *cipher* in *ENCRYPT_MODE* initialisiert...

```
private void initEnc(){
  // initialise the cipher for encryption
  try {
    cipher.init( Cipher.ENCRYPT_MODE, publicKey,
      systemParameters, new SecureRandom() );
  }
  catch( InvalidKeyException ike ) {
    // handle exception
    ike.printStackTrace();
  }
  catch( InvalidAlgorithmParameterException iape ) {
    // handle exception
    iape.printStackTrace();
  }
}
```

Quellcode 8.6: Verschlüsseln initialisieren

und zum Verschlüsseln der Daten eingesetzt.

```
private void encrypt(){
  // encrypt the plaintext
  // byte plaintext[]=new String("secret message").getBytes();
  // or
  byte plaintext[] = readFile("plaintextFile");

  ciphertext = new byte[0];
  try {
    ciphertext = cipher.doFinal( plaintext );
  }
  catch( IllegalBlockSizeException ibse ) {
    // handle exception
    ibse.printStackTrace();
  }
  catch( BadPaddingException bpe ) {
    // handle exception
    bpe.printStackTrace();
```

```
  }

  writeFile("ciphertextFile", ciphertext);
}
```

Quellcode 8.7: Verschlüsseln der Daten

Analog zum öffentlichen Schlüssel wird der zur Identität passende private Schlüssel generiert. Dafür werden aber der *masterKey* und die *map* zusätzlich benötigt.

```
private void loadPrivateKey(){
  // initialise the key pair generator
  keyParameters = new IbeKeyParameters( hash, identity,
    masterKey, map );
  try {
    kpg.initialize( keyParameters );
  }
  catch( InvalidAlgorithmParameterException iape ) {
    // handle exception
    iape.printStackTrace();
  }
  keyPair = kpg.generateKeyPair();
  privateKey = keyPair.getPrivate();
}
```

Quellcode 8.8: Privater Schlüssel laden

Zum Entschlüsseln wird *cipher* in *DECRYPT_MODE* initialisiert...

```
private void initDec(){
  // initialise the cipher for decryption
  try {
    cipher.init( Cipher.DECRYPT_MODE, privateKey,
    systemParameters, new SecureRandom() );
  }
  catch( InvalidKeyException ike ) {
    // handle exception
    ike.printStackTrace();
  }
  catch( InvalidAlgorithmParameterException iape ) {
    // handle exception
    iape.printStackTrace();
  }
}
```

Quellcode 8.9: Entschlüsseln initialisieren

und mit der Funktion *doFinale*() der verschlüsselte Text entschlüsselt.

```
private void decrypt(){
  byte ciphertext[] = readFile("ciphertextFile");

  // decrypt the ciphertext
  byte decryptedText[] = new byte[0];
  try {
```

```
    decryptedText = cipher.doFinal( ciphertext );
  }
  catch( IllegalBlockSizeException ibse ) {
    // handle exception
    ibse.printStackTrace();
  }
  catch( BadPaddingException bpe ) {
    // handle exception
    bpe.printStackTrace();
  }

  writeFile("decryptedtextFile", decryptedText);
}
```

Quellcode 8.10: Entschlüsseln der Daten

Die Funktionen *readFile*() bzw. *writeFile*() werden hier nicht erläutert, sie dienen nur zum Lesen bzw. Schreiben der Daten aus bzw. in einer Datei.

Der Einsatz des IBE-JCA Providers zum Signieren bzw. zur Signatur-Prüfung unterscheidet sich programmiertechnisch nicht so sehr von dem zum Ver- bzw. Entschlüsseln. Es kommen weitere Klassen zum Einsatz und einige Objekte werden mit anderen Optionen initialisiert.

```
provider = new BioIbsProvider();
//statt
provider = new IbeProvider();

signature = Signature.getInstance( "BioIbs", provider );
//statt
cipher = Cipher.getInstance( "ibe", provider );

kpg = KeyPairGenerator.getInstance( "BioIbs", provider );
//statt
kpg = KeyPairGenerator.getInstance( "ibe", provider );

systemParameters =
  new BioIbsSystemParameters(map,hash,masterKey,ecc );
//statt
systemParameters =
  new IbeSystemParameters(map,hash,masterKey );

keyParameters =
  new IbeKeyParameters(hash,
    identity,masterKey,map);
//statt
keyParameters =
  new BioIbsKeyPairGeneratorParameters(hash,
    identity,masterKey,map);
```

Quellcode 8.11: Unterschied Signatur-Verschlüsseln

Zum Signieren eines *message* werden folgende Befehle ausgeführt:

```
KeyPair keyPair = kpg.generateKeyPair();

PrivateKey privateKey = keyPair.getPrivate();
PublicKey publicKey = keyPair.getPublic();

BioIbsPrivateKey privKey=(BioIbsPrivateKey)privateKey;
BioIbsPublicKey pubKey=(BioIbsPublicKey)publicKey;

try {
  // sign the document
  signature.initSign( privateKey );
  signature.update( message );
  sig = signature.sign();
}
catch( InvalidKeyException ike ) {
  System.err.println( ike );
}
catch( SignatureException se ) {
  System.err.println( se );
}
```

Quellcode 8.12: Signieren einer Nachricht

Hier erfolgt die Prüfung der Signatur:

```
try {
  // verify the signature
  signature.initVerify( publicKey );
  signature.update( message );
  if( signature.verify( sig ) ) {
      //Signature verified!
  }
  else {
      //Signature not verified!
  }
}
catch( InvalidKeyException ike ) {
  System.err.println( ike );
}
catch( SignatureException se ) {
  System.err.println( se );
}
```

Quellcode 8.13: Signatur prüfen

## 8.3 Bild als Identität

Ein Bild als Identität in manchen Szenarien einzusetzen scheint sinnvoll zu sein (s. Kapitel 7). Ist dies aber machbar? Ist ein Bild als Identität in der *Identitätsbasierten Kryptographie*, technisch gesehen, geeignet? Um das zu prüfen sind eine Reihe Ver-

suche durchgeführt worden, die ermöglicht haben, den Einfluss der Identitätslänge auf das System genauer zu untersuchen.

Es wurden Dateien mit aus einer grossen Bilddatei stammenden Bytes in verschiedenen Grössen generiert, 27 Byte, 1, 10, 100 KByte, 1 und 10 MByte. Diese Dateien wurden dann als Identität geladen und zum Ver- bzw. Entschlüsseln einer Datei benutzt. Die Tabelle 8.2 stellt eine Zusammenfassung einiger Messungen dar.

In der zweiten Versuchsreihe wurden die generierten Dateien mit der selben Identität (27 Byte groß) ver- bzw. entschlüsselt. Der Einfluss der Grösse der zu verschlüsselnden Daten auf das System ist in der Tabelle 8.1 zusammengefasst.

Die Versuche lassen folgendes erkennen:

- „Identitäten“ bis einer Länge (Größe) von 10 MByte lassen sich problemlos einsetzten. Einsatz von größeren Bilder (mehr als 10 MByte) ist sinnlos, denn es werden Mobile Geräte zum Darstellen bzw. Prüfen benutzt, die meistens nur begrenzte Resourcen zur Verfügung stellen und mit solchen Dateien überfordert werden könnten. 10 MByte Dateien sind auch übertrieben groß und wurden nur als Bestätigung der Ergebnissen geprüft.
- Das Initialisieren des Systems hängt nur vom System ab. Die Identitätslänge und die Datenmenge haben keinen Einfluss darauf.
- Einen Zusammenhang zwischen dem Initialisieren der Systemparameter und der Identitätslänge bzw. Datenmenge ist auch nicht festzustellen. Warum die Werte so stark variieren ließ sich mit den Versuchen nicht erklären.
- Ein Einfluss der Länge der Identität und der Größe der Datenmenge ist nur auf die zum Lesen bzw. Schreiben (Zugriff auf dem Datenträger) benötigten Zeiten zu merken.
- Die verschlüsselten Daten sind um etwa 1028 Byte grösser als der Klartext und dies ist unabhängig sowohl von der Länge der Identität als auch von der Datenmenge.
- Das Ver-, Entschlüsseln und das Laden der Identität [1] dauern relativ lange (mehr als 10 Sekunden). Alle weiteren Werte halten sich in Grenzen (max. 3 Sekunden). Das unterscheidet dieses System nicht von anderen kryptographischen Systemen.

Wie die Versuche zeigen, ist der Einsatz eines Bildes in einem identitätsbasierten Verschlüsselungssystem machbar. Es ist jedoch noch zu prüfen, wie zuverlässig das bildbasierte System ist.

[1] besonders bei sehr langen Identitäten

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 94 | 1625 | 0 | 27 | 0 | 0 | 1024 | 15547 | 0 | 2052 | 2906 | 16 | 12312 | 0 |
| 31 | 531 | 1422 | 15 | 27 | 0 | 0 | 1024 | 15469 | 16 | 2052 | 2766 | 0 | 12468 | 0 |
| 32 | 0 | 1453 | 0 | 27 | 0 | 0 | 1024 | 15719 | 0 | 2051 | 2828 | 0 | 12594 | 0 |
| 47 | 0 | 1391 | 0 | 27 | 0 | 15 | 10240 | 15719 | 31 | 11267 | 2765 | 16 | 12813 | 31 |
| 47 | 4516 | 1406 | 16 | 27 | 0 | 16 | 10240 | 15531 | 16 | 11268 | 2906 | 16 | 12312 | 16 |
| 47 | 359 | 1343 | 0 | 27 | 0 | 16 | 10240 | 15437 | 32 | 11268 | 2781 | 15 | 12610 | 31 |
| 46 | 0 | 1407 | 0 | 27 | 0 | 141 | 102400 | 15547 | 250 | 103426 | 2813 | 140 | 12453 | 250 |
| 31 | 15 | 1375 | 0 | 27 | 0 | 140 | 102400 | 15250 | 266 | 103428 | 2891 | 140 | 12188 | 234 |
| 31 | 1156 | 1469 | 0 | 27 | 0 | 141 | 102400 | 15844 | 250 | 103428 | 2907 | 141 | 12531 | 250 |
| 31 | 9079 | 1547 | 0 | 27 | 0 | 1515 | 1048576 | 15969 | 2562 | 1049604 | 2796 | 1469 | 12343 | 2500 |
| 31 | 16 | 1406 | 0 | 27 | 0 | 1437 | 1048576 | 15609 | 2563 | 1049603 | 2922 | 1469 | 12391 | 2500 |
| 47 | 922 | 1359 | 0 | 27 | 0 | 1469 | 1048576 | 15297 | 2531 | 1049604 | 2735 | 1438 | 12140 | 2485 |
| 32 | 531 | 1484 | 0 | 27 | 0 | 1532 | 1048576 | 15890 | 2547 | 1049604 | 2719 | 1484 | 12094 | 2516 |
| 47 | 0 | 1297 | 0 | 27 | 0 | 1469 | 1048576 | 14500 | 2531 | 1049604 | 2734 | 1469 | 11516 | 2515 |
| 47 | 0 | 1297 | 0 | 27 | 0 | 1438 | 1048576 | 14547 | 2531 | 1049604 | 2672 | 1453 | 11734 | 2516 |
| 32 | 2984 | 1344 | 15 | 27 | 0 | 14579 | 10485760 | 14609 | 25109 | 10486788 | 2532 | 14703 | 11140 | 24860 |
| 47 | 0 | 1422 | 0 | 27 | 0 | 14515 | 10485760 | 13860 | 25000 | 10486788 | 2500 | 14734 | 10922 | 24906 |
| 32 | 0 | 1281 | 0 | 27 | 0 | 14610 | 10485760 | 13890 | 25110 | 10486788 | 2594 | 14703 | 10859 | 24953 |

(A) System initialisieren (ms)
(B) Systemparameter initialisieren (ms)
(C) Schlüsselparameter erstellen (ms)
(D) Identität aus Datei laden (ms)
(E) Identitätslänge (Byte)
(F) Identität laden (ms)
(G) Klartext aus Datei laden (ms)
(H) Klartextlänge (Byte)
(I) Verschlüsseln (ms)
(J) Verschlüsselter Text in Datei schreiben (ms)
(K) „Verschlüsselter Text“-Länge (Byte)
(L) Privater Schlüssel generieren (ms)
(M) Verschlüsselter Text aus Datei laden (ms)
(N) Entschlüsseln (ms)
(O) Entschlüsselter Text in Datei schreiben (ms)

Tabelle 8.1: Einfluss der Datenmenge

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 8407 | 1609 | 0 | 27 | 0 | 16 | 1024 | 15640 | 16 | 2052 | 2781 | 0 | 12297 | 16 |
| 31 | 0 | 1375 | 16 | 27 | 0 | 0 | 1024 | 15531 | 16 | 2052 | 2828 | 16 | 12343 | 16 |
| 31 | 1485 | 1375 | 0 | 27 | 0 | 0 | 1024 | 15453 | 15 | 2052 | 2750 | 16 | 12359 | 0 |
| 45 | 94 | 1484 | 0 | 1024 | 0 | 16 | 1024 | 15344 | 0 | 2052 | 2797 | 0 | 12078 | 0 |
| 47 | 469 | 1313 | 0 | 1024 | 0 | 0 | 1024 | 15375 | 15 | 2052 | 2797 | 0 | 12234 | 0 |
| 47 | 0 | 1344 | 16 | 1024 | 0 | 0 | 1024 | 14906 | 16 | 2051 | 2718 | 0 | 12016 | 16 |
| 31 | 656 | 1375 | 16 | 10240 | 0 | 0 | 1024 | 15343 | 0 | 2052 | 2750 | 16 | 12219 | 0 |
| 47 | 1562 | 1344 | 16 | 10240 | 0 | 0 | 1024 | 14875 | 15 | 2051 | 2688 | 0 | 11906 | 0 |
| 47 | 0 | 1328 | 16 | 10240 | 0 | 15 | 1024 | 15344 | 16 | 2051 | 2750 | 15 | 12219 | 16 |
| 31 | 2219 | 1406 | 141 | 102400 | 31 | 0 | 1024 | 15141 | 15 | 2052 | 2891 | 16 | 12093 | 16 |
| 31 | 1750 | 1406 | 141 | 102400 | 32 | 15 | 1024 | 14891 | 0 | 2052 | 2750 | 0 | 11937 | 0 |
| 47 | 9000 | 1406 | 156 | 102400 | 31 | 0 | 1024 | 14907 | 0 | 2052 | 2797 | 0 | 11906 | 15 |
| 32 | 9093 | 1344 | 1610 | 1048576 | 93 | 0 | 1024 | 15140 | 0 | 2051 | 2985 | 0 | 11984 | 0 |
| 47 | 3844 | 1265 | 1579 | 1048576 | 93 | 0 | 1024 | 15203 | 0 | 2052 | 2704 | 0 | 12187 | 16 |
| 31 | 3312 | 1329 | 1562 | 1048576 | 94 | 15 | 1024 | 14844 | 0 | 2051 | 2797 | 0 | 11969 | 15 |
| 32 | 4516 | 1281 | 14953 | 10485760 | 438 | 0 | 1024 | 13843 | 16 | 2052 | 2844 | 0 | 11062 | 16 |
| 31 | 719 | 1250 | 14921 | 10485760 | 422 | 0 | 1024 | 13532 | 0 | 2051 | 2875 | 15 | 10797 | 0 |
| 47 | 0 | 1281 | 14797 | 10485760 | 428 | 0 | 1024 | 13688 | 15 | 2051 | 2829 | 0 | 10953 | 0 |

| | | |
|---|---|---|
| (A) System initialisieren (ms) | (B) Systemparameter initialisieren (ms) | (C) Schlüsselparameter erstellen (ms) |
| (D) Identität aus Datei laden (ms) | (E) Identitätslänge (Byte) | (F) Identität laden (ms) |
| (G) Klartext aus Datei laden (ms) | (H) Klartextlänge (Byte) | (I) Verschlüsseln (ms) |
| (J) Verschlüsselter Text in Datei schreiben (ms) | (K) „Verschlüsselter Text“-Länge (Byte) | (L) Privater Schlüssel generieren (ms) |
| (M) Verschlüsselter Text aus Datei laden (ms) | (N) Entschlüsseln (ms) | (O) Entschlüsselter Text in Datei schreiben (ms) |

Tabelle 8.2: Einfluss der Schlüssellänge

## 8.4 Zuverlässigkeit

Um die Zuverlässigkeit des Systems zu testen, ist eine Applikation entwickelt worden, die folgendermassen vorgeht:

1. Die „Identität“ $ID$ laden und ein Schlüsselpaar $(P, S)$ generieren,
2. Ein zufälliges Byte aus der „Identität“ mit einem zufälligen Wert ersetzen. Aus der manipulierten „Identität“ $ID'$ wird ein Schlüsselpaar $(P', S')$ generiert,
3. Eine Nachricht mit dem privaten Schlüssel $S$ signieren,
4. Die Signatur mit dem öffentlichen Schlüssel $P$ prüfen,
5. Die Signatur mit dem „manipulierten “ öffentlichen Schlüssel $P'$ prüfen,
6. Ergebnisse speichern.

Die Applikation ist hundert mal mit verschiedenen Identitäten durchgeführt worden und lieferte das eindeutige Ergebnis: Die Prüfung mit dem $P$ war immer erfolgreich, dafür lieferte $P'$ bei allen hundert Tests keine Übereinstimmung. D.h. eine Abweichung eines einzigen Bytes der Identität erzeugt ein völlig anderes Schlüsselpaar. Somit ist die Fälschung solcher Ausweise kaum möglich.

Will man jedoch eine Toleranz in dem System einbauen, d.h. kleine Abweichungen in dem Bild sollen die Prüfung der Signatur nicht verhindern, kann man den sogenannten „fuzzy extractor“ benutzen. Der „fuzzy extractor“ generiert aus biometrischen Daten (hier das Bild) eine Zeichenkette, die als Identität in dem System benutzt werden kann.

Dieses Zitat aus [DORS07, Seite 2] erklärt, was ein „fuzzy extractor“ ist. Weitere Informationen wie die Funktionsweise und die zugrunde liegenden mathematischen Funktionen kann man dem Paper [DORS07] entnehmen.

> ..., an important general problem is to convert noisy nonuniform inputs into reliably reproducible, uniformly random strings. To this end, we propose a new primitive, termed fuzzy extractor. It extracts a uniformly random string $R$ from its input $w$ in a noise-tolerant way. Noise tolerance means that if the input changes to some $w'$ but remains close, the string $R$ can be reproduced exactly. To assist in reproducing $R$ from $w'$, the fuzzy extractor outputs a non-secret string $P$. It is important to note $R$ remains uniformly random even given $P$.

# Kapitel 9

# Implementierung des Systems

Das System besteht aus zwei bzw. drei Komponenten, je nachdem ob die Tickets auf einem Server gespeichert werden müssen oder auf dem mobilen Gerät des Kunden. Die Komponenten werden in dieser Arbeit wie folgt bezeichnet:

***PbAE*-Server** Auf dem Server werden die Tickets, falls erwünscht, gespeichert

***PbAE*-Client** ist das Gerät, was der Kontrolleur benutzt oder welches zum Ausstellen des Tickets zum Einsatz kommt

***PbAE*-Mobile** ist das mobile Kundengerät, es kann Handy oder PDA bzw. MDA sein.

Die Kommunikation zwischen dem ***PbAE***-Client und dem ***PbAE***-Server läuft über konventionelle Netzwerkverbindung (LAN, WAN oder WLAN). Die Implementierung solcher auf Socket-basierte Verbindung stellte keine Herausforderung dar und wird in dieser Arbeit nicht genauer erläutert.

Die Verbindung zwischen dem ***PbAE***-Client und dem ***PbAE***-Mobile kann kabelgebunden oder kabellos sein. Für die kabelgebunden Lösung hätte der ***PbAE***-Client mit Adaptern für die verschiedenen mobilen Geräte ausgestattet sein sollen. Deswegen kam für uns nur eine kabellose Verbindung in Frage und wir haben uns für die Verbindung über die Bluetooth-Schnittstelle entschieden, denn sie ist stabiler und flexibler als die über die Infrarot-Schnittstelle.

Die Implementierung dieser ***PbAE***-Client - ***PbAE***-Mobile Kommunikation wird in diesem Abschnitt behandelt.

## 9.1 Implementierung

### 9.1.1 *PbAE*-Server

Wie oben erläutert, kommt ein ***PbAE***-Server nur zum Einsatz nur, wenn die Tickets auf dem Server gespeichert werden müssen. Der ***PbAE***-Server wurde in J2SE geschrieben. Er soll als Dämon-Prozess im Hintergrund laufen, deswegen wurde auf eine graphische Oberfläche verzichtet.

Auf die Funktionsweise des ***PbAE***-Servers wird hier nicht näher eingegangen. Zusammengefasst: es werden `DataInputStream` und `DataOutputStream` zur Kommunikation mit dem ***PbAE***-Client eingesetzt. Die Verbindung kann verschlüsselt aufgebaut werden. Der Zugriff auf die MySQL Datenbank erfolgt über den JDBC-Treiber JConnector von MySQL.

### 9.1.2 *PbAE*-Client

Der ***PbAE***-Client muss evtl. gleichzeitig mit den beiden anderen Komponenten kommunizieren können. Er besteht aus einer graphischen in Swing programmierten Oberfläche mit vier Hauptkomponenten:

**Bedienelemente:** Sind die Menüs und die Konfigurationsseiten.

**Canvas-Komponente:** Zur Darstellung des Bildes.

**Net-Komponente:** Zur evtl. Kommunikation mit dem Server.

**Bluetooth-Komponente:** Zur Kommunikation mit dem mobilen Gerät.

Die drei ersten Komponenten ließen sich problemlos implementieren, die Bluetooth-Komponente war etwas komplizierter zu implementieren.

#### 9.1.2.1 Bluetooth-Komponente

Die Entwicklungsumgebung J2SE bietet keine Unterstützung der Programmierung der Bluetooth-Schnittstelle an, daher war eine zusätzliche Bibliothek erforderlich, die die Bluetooth-Geräte über Java ansprechbar machen kann. Für diese Applikation ist die Wahl auf das Bluetooth Development Kit von Avetana gefallen, wegen ihrer zahlreichen Funktionen und der sehr guten Betreuung des Herstellers, und vor allem wegen der Kompatibilität zur vorhandenen Hardware.

Der Dienstleister ist mit einer *SID* (Service-ID) identifizierbar, die dem Kunden bei der Anmeldung mitgeteilt werden muss.

Bei Bedarf, wird die Bluetooth-Schnittstelle initialisiert und gestartet. Sie lauscht auf Verbindungsanfragen.

```
...
/*Hardwareerkennung*/
localDevice = localDevice.getLocalDevice();

/*Aus der Dienst-ID eine UUID Instanz generieren*/
SID = new UUID(serviceID, false);

/*Die URL der Verbindung bilden*/
this.url="btspp://localhost:" + SID.toString() +//
          ";name=PBAE-Client";
...

/*Dienst starten*/
service = (StreamConnectionNotifier) //
```

```
            Connector.open(this.url);

/*Alle verfügbaren Dienste herausfinden*/
ServiceRecord record = localDevice.getRecord(service);

/*Gerät für die Anderen sichtbar machen*/
localDevice.setDiscoverable(DiscoveryAgent.GIAC);
...
```

Quellcode 9.1: Initialisierung und Starten des Bluetoothgerätes

Das hier verwendete Protokoll ist *bt_spp*, das auf das Profil *SPP* (Serial Port Profile) aufbaut. Mit diesem Profil können serielle Kabelverbindungen (RS-232) emuliert werden. Es muss gemäß des Standards eine Übertragungsrate von bis zu 128 kb/s unterstützen.

Wenn eine Anforderung an den Service vorhanden ist, wird die Verbindung akzeptiert und der Kanal mittels der Methode `acceptAndOpen()` geöffnet. Aus der zurückgelieferten `StreamConnection` werden Ein- bzw. Ausgabekanäle erzeugt, die zur Datenübertragung eingesetzt werden.

```
...
/*Verbindung akzeptieren und Kanal öffnen*/
conn = service.acceptAndOpen();

/*Gerät für die Anderen sichtbar machen*/
input = new DataInputStream(conn.openInputStream());
output = new DataOutputStream(conn.openOutputStream());
...
```

Quellcode 9.2: Aufbau der Bluetooth-Verbindung

Wie eine verschlüsselte Bluetooth-Verbindung aufgebaut werden kann, kann [Tho07] entnommen werden.

### 9.1.3 *PbAE*-Mobile

Bevor auf die Implementierung des ***PbAE***-Mobile eingegangen wird, werden ein paar Begriffe erläutert.

#### 9.1.3.1 J2ME

*J2ME*, vor kurzem in *JavaME* umbenannt, ist eine abgespeckte Version der Programmiersprache Java, die es erlaubt, Java-Applikationen auf mobilen Geräte zum Laufen zu bringen. Die begrenzten Ressourcen der mobilen Geräte, wie Mobiltelefone oder PDAs, reichen nicht aus für die hohen Anforderungen der Standardversion von Java (Java SE oder auch J2SE genannt).(Abbildung 9.1, Seite 73)

#### 9.1.3.2 Konfiguration

In der Konfiguration sind Java-Klassen definiert, welche für eine Geräte-Klasse zur Verfügung stehen sollen. In Java ME sind zwei Konfigurationen verfügbar: **CDC**

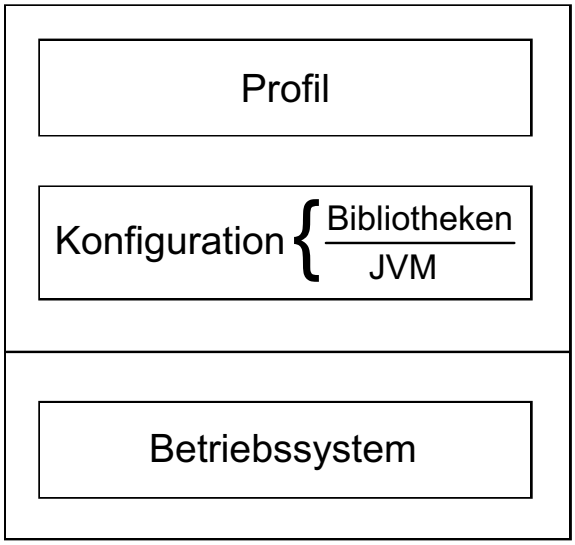



Abbildung 9.1: J2ME-Architektur

und **CLDC**. Sie geben auch die Minimalanforderungen des Systems an.

**CDC** Die Connected Device Configuration richtet sich bei einem Speicherbedarf von etwa 2 MB an vergleichsweise leistungsfähige Geräte. Dafür muss die JVM der CDC laut Spezifikation alle Fähigkeiten der JVM von Java SE besitzen. Aufgrund der hohen Speicheranforderungen ist die CDC jedoch nicht auf Mobiltelefonen sondern nur auf leistungsfähigeren Modellen wie Smart Phones oder PDAs lauffähig.

**CLDC** Die Connected Limited Device Configuration hingegen beschreibt eine minimalistische Umgebung für äußerst ressourcenarme Geräte. In ihrer Spezifikation von SUN MICROSYSTEMS, INC. wird das Ziel der CLDC beschrieben [Mic]:

> „The main goal of the CLDC Specification is to standardize a highly portable, minimum-footprint Java™ application development platform for resourceconstrained, connected devices.“

Das ***PbAE***-Mobile soll auf allen mobilen Geräte lauffähig sein, deswegen ist die CLDC für diese Arbeit die gewählte Konfiguration.

Aus der Java SE wurden Untermengen von Klassen definiert, die in der CLDC zur Verfügung stehen müssen. Außerdem sind CLDC-spezifische Klassen entwickelt worden, insbesondere für Ein-, Ausgabe und Kommunikation.

Das *Generic Connection Framework (GCF)* ist CLDC-spezifisch und stellt eine generalisierte Form der Ein-/Ausgabe-Klassen von Java SE dar. Ziele des GCF sind u.a. eine bessere Erweiterbarkeit und Flexibilität. Durch das GCF ist es einfach möglich, neue Protokolle oder Geräte zu unterstützen.
Mit Hilfe der Methode

```
Connector.open("<protocol>:<address>;<parameters>");
```

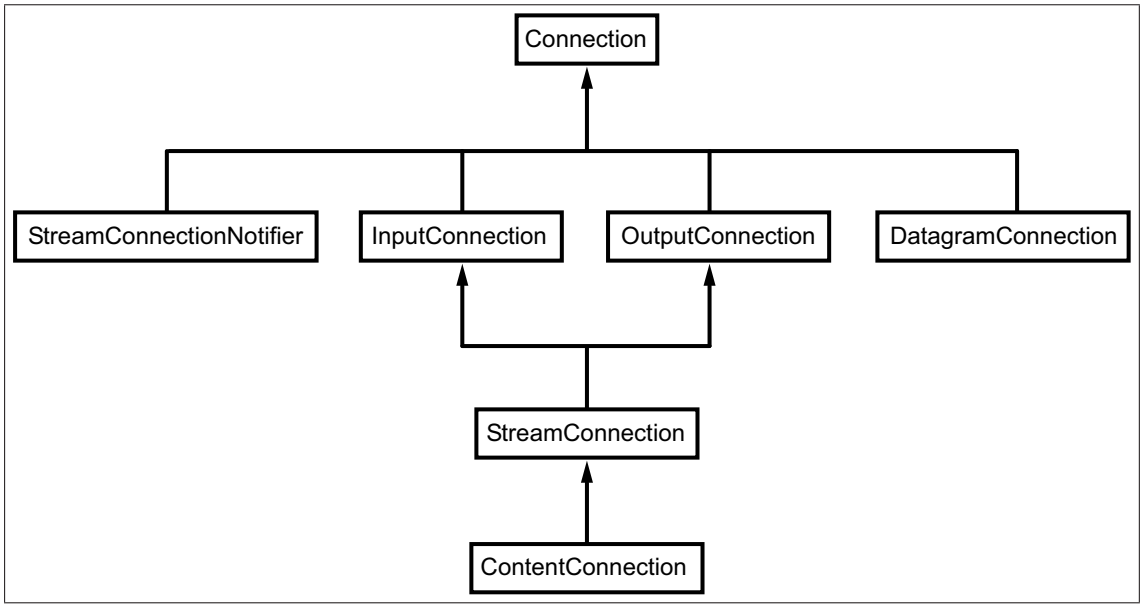



Abbildung 9.2: Vererbungshierarchie der Interfaces des GCF

kann für beliebige Protokolle eine Verbindung aufgebaut werden.
Da CLDC selbst aufgrund der diversen Gerätetypen keine Implementierung konkreter Protokolle bietet, besteht das in der CLDC spezifizierte GCF nur aus Interfaces. In Abbildung 9.2 (Seite 74) ist die Vererbungsstruktur der Interfaces des GCF veranschaulicht. Diese Interfaces können in auf CLDC aufbauenden Profilen für konkrete Verbindungsprotokolle implementiert werden.

#### 9.1.3.3 Profile

Darüberhinaus werden Klassen und Features, die auf bestimmten Aufgaben oder Geräte-Klassen zugeschnitten sind, in Profilen definiert, die auf Konfigurationen basieren. Zur Zeit sind zwei auf CLDC basierende Profile verfügbar:

**IMP** Information Module Profile richtet sich an Geräte ohne Benutzerschnittstelle, daher irrelevant für diese Arbeit.

**MIDP** Mobile Information Device Profile richtet sich an mobilen Geräte mit Eingabeschnittstellen wie Tastatur oder Touchpad, sowie Ausgabeschnittstellen wie Bildschirme, und kabelgebundene oder kabellose Netzwerkschnittstellen.

Für das ***PbAE***-Mobile ist die CLDC-Konfiguration mit dem Profile MIDP 2.0 erforderlich.

#### 9.1.3.4 JSR

Ein „Java Specification Request“ (JSR) ist eine Anforderung einer neuen Java-Spezifikation oder einer wichtigen Änderung einer existierenden Java-Spezifikation. Über JSR werden neue Java-Standards definiert und auch Erweiterungen der Programmiersprache Java oder der Java-Laufzeitumgebung gemeinschaftlich entwickelt. Die für diese Arbeit erforderlichen Standards sind:

**JSR-139** Unterstützung der Konfiguration CLDC 1.1

**JSR-118** Unterstützung des Profils MIDP 2.0

**JSR-75** Lese- und Schreibzugriff auf das lokale Dateisystem

**JSR-82** Unterstützung der Steuerung von Bluetooth-Schnittstellen

Diese Spezifikationen müssen von der Hardware unterstützt werden und die vom Hardware-Hersteller implementierte Java-Laufzeitumgebung muss dies auch anbieten, was erst bei den mobilen Geräte der neueren Generationen der Fall ist.

Die Handy- bzw. PDA-Hersteller richten sich meistens an Endverbraucher, weswegen Informationen über die unterstützten Spezifikationen selten Bestandteil der Handbücher sind. Auf der Homepage der Hersteller bin ich auch selten fündig geworden. Nur der technische Support der Hersteller oder einige Test- und Vergleichseiten im Internet wie http://www.areamobile.de verraten solche für Softwareentwickler relevanten Daten.

Nach langer Recherche ist die Wahl auf die Geräte Nokia N70 und das Sony Ericsson K800i als Testgeräte gefallen. Die zwei Modelle waren einige der ersten, die JSR-82 und JSR-75 erfüllt haben. Mittlerweile kommen fast alle neuen Modelle in Frage.

#### 9.1.3.5 Implementierung des MIDlet

Das MIDlet[1] ist in der Entwicklungsumgebung „Sun Java wireless Toolkit 2.5 for CLDC“ entwickelt worden. Es besteht aus drei Hauptbausteinen:

**Grafischen Oberfläche** Zur Steuerung und Konfiguration der Applikation. Dafür sind die von Java ME gebotene Funktionen ausreichend.

**Datei-Lese- und Schreibmodule** Erlauben das Lesen bzw. Schreiben von Dateien. Die JSR-75-Klassen sind dafür erforderlich. Für die Bearbeitung von XML-Dateien, wie die Konfigurationsdateien, werden außerdem zusätzliche Bibliotheken benötigt. In dieser Arbeit ist kXML2 eingesetzt worden.

**Bluetooth-Modul** ermöglicht die Kommunikation über die Bluetooth-Schnittstelle. Die Programmierung geschieht anhand der JSR-82-Klassen.

Damit die Applikation steuerbar bleibt, auch während der Lese-, Schreib- und Datenübertragungsvorgänge, sind die Module als nebeneinander lauffähige Threads implementiert worden.

Bei der ersten Anmeldung bei einem Dienstleister werden Daten empfangen und in einer XML-Datei eingetragen, welche die folgende Struktur hat:

```
<?xml version="1.0" encoding="UTF-8"?>
<pbaem:services xmlns:pbaem="http://www.pbae.org/pbaem/">
  <pbaem:service>
    <pbaem:name>Deutsche Bahn AG</pbaem:name>
```

[1] Eine auf Mobiltelefonen oder vergleichbaren mobilen Geräten lauffähige Software, welche in Java geschrieben ist und dem MIDP entspricht

```
    <pbaem:sid>F0E...011</pbaem:sid>
    <pbaem:auth_key>datei1</pbaem:auth_key>
  </pbaem:service>
  <pbaem:service>
    <pbaem:name>AOK</pbaem:name>
    <pbaem:sid>F0E...010</pbaem:sid>
    <pbaem:auth_key>datei2</pbaem:auth_key>
  </pbaem:service>
</pbaed:services>
```

Quellcode 9.3: Dienste-Datei

Wie diese Datei heißt und wo sie sich befindet, wird in einer Konfigurationsdatei angegeben, die beim ersten Start der Applikation nach der Installation angelegt wird. Die Konfigurationsdatei gibt auch an, wo die Benutzerdaten und die LOG-Datei zu finden sind.

```
<?xml version="1.0" encoding="UTF-8"?>
<pbaem:xml_config xmlns:pbaem="http://www.pbae.org/pbaem/">
  <pbaem:logFile path="/E:/pbaem/pbaem.log"/>
  <pbaem:servicesFile path="/E:/pbaem/services.xml"/>
  <pbaem:keysDir path="/E:/pbaem/keys/"/>
</pbaem:xml_config>
```

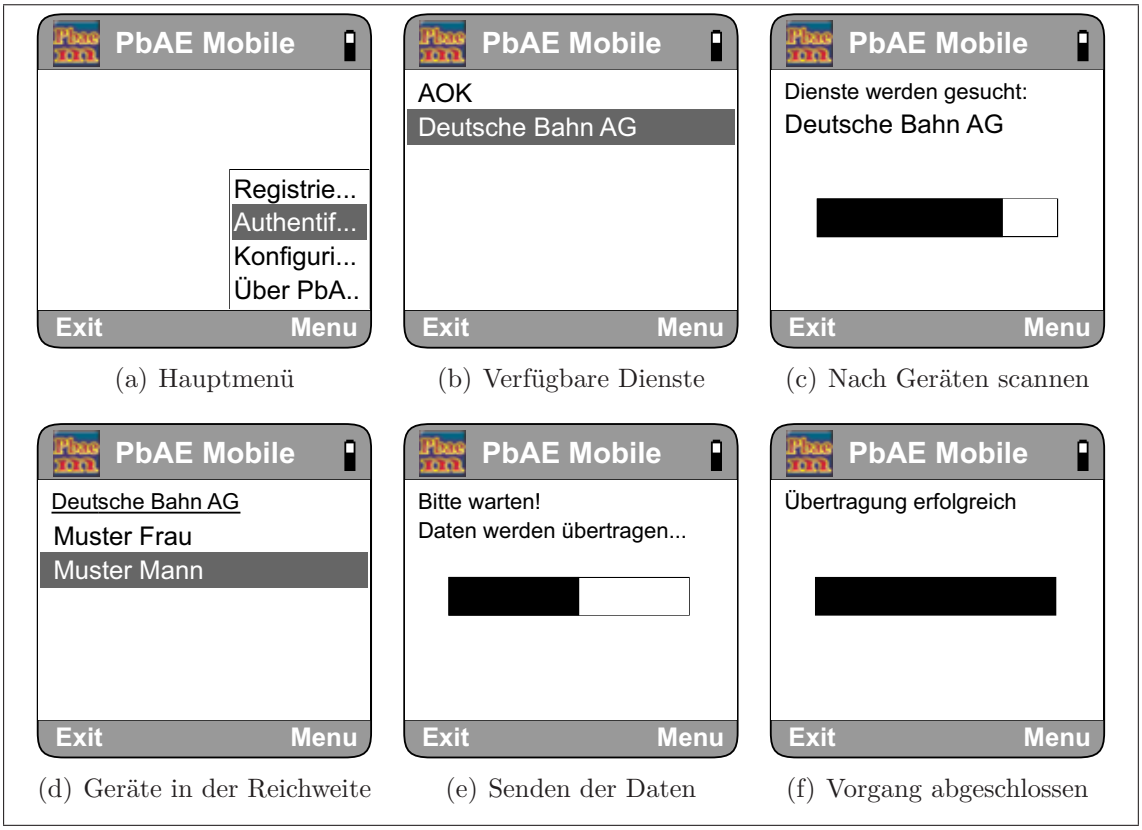
Quellcode 9.4: Konfigurationsdatei des ***PbAE***-Mobiles

Nach dem Parsen der Datei ist die Anwendung bereit zum Einsatz. Möchte der Besitzer sich mittels des ***PbAE***-Systems authentifizieren, werden zuerst die in der Services-Datei angegebenen Dienstleistungsanbieter dargestellt. (Abbildung 9.3(b) (Seite 77)

Nach der Wahl des Dienstes, wird nach der dazugehörigen *SID* in der Services-Datei gesucht.
Nach der Initialisierung der Bluetooth-Schnittstelle werden alle verfügbaren Bluetoothgeräte in der Reichweite gesucht und geprüft, ob sie Dienste mit der *SID* anbieten (Abbildung 9.3(c) (Seite 77). Wenn ja, dann werden sie dem Benutzer als Auswahlliste dargestellt (Abbildung 9.3(d) (Seite 77).

```
  ...
  // Bluetooth initialisieren
  LocalDevice localDevice = LocalDevice.getLocalDevice();
  localDevice.setDiscoverable(DiscoveryAgent.GIAC);
  discoveryAgent = localDevice.getDiscoveryAgent();
  ...
  // Suche nach Bluetoothgeräte in der Reichweite starten
  discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);

  // Sind welche gefunden, wird geprüft ob sie die
  // Dienste mit der sid anbieten
  // rd: ist das gefundene Gerät (Remote Device)
  ...
  uuidSet = new UUID[1];
  uuidSet[0] = new UUID(sid, false);
```




(a) Hauptmenü (b) Verfügbare Dienste (c) Nach Geräten scannen

(d) Geräte in der Reichweite (e) Senden der Daten (f) Vorgang abgeschlossen

Abbildung 9.3: ***PbAE*** Mobile

```
searchIDs[i] = discoveryAgent.searchServices(null,
                    uuidSet, rd, this);
...
```

Quellcode 9.5: Mobile Bluetoothschnittstelle

Entscheidet sich der Benutzer für einen davon, werden die Kommunikationskanäle aufgebaut.

```
...
// Die url abfragen.
url=service.getConnectionURL(...);
...
// Die Verbindung aufbauen
conn = (StreamConnection) Connector.open(url);
            rd, this);
...
// Output- und Inputstream öffnen
output = new DataOutputStream(conn.openOutputStream());
input = new DataInputStream(conn.openInputStream());
...
```

Quellcode 9.6: Bluetoothverbindung aufbauen (Mobile)

Erfolgt die Verbindung, und sind die Kanäle aufgebaut worden, dann kann die Kommunikation stattfinden (Abbildung 9.3(e), Seite 77) und eine Authentifizierung und Ticket-Bestellung bzw. -Prüfung durchgeführt werden.

# Kapitel 10

# Technische und finanzielle Aspekte

Bei der Entwicklung neuer Systeme sind auch die technischen und finanziellen Anforderungen zu betrachten. Am Anfang dieser Arbeit lag die größte Schwierigkeit darin, die geeignete Software und Hardware zu finden. Im Laufe der Jahre hat sich die Lage verbessert, besonders was die Hardware betrifft.

Serverseitig sind „Standard"-Rechner mit genügend[1] Resourcen und einer Netzwerkschnittstelle ausreichend. Solche Server sind von namhaften Herstellern schon für unter 1.000,- Euro zu bekommen.

Der Client muss eine Bluetooth-Verbindung zum mobilen Gerät herstellen können, was mit weiteren Kosten verbunden sein kann. Falls der Client über keine eingebaute Bluetooth-Schnittstelle verfügt, kann er mit einer externen erweitert werden. Es ist nur darauf zu achten, dass die Hardware von der Software unterstützt wird. Für meine Arbeit ist ein Bluetooth USB-Stick der Marke Belkin eingesetzt worden. Dieser ist für ca. 20,- Euro angeschafft worden.

Um die Steuerung der Bluetooth-Schnittstelle über Java zu ermöglichen, ist eine zusätzliche Bibliothek erforderlich (siehe Abschnitt 9.1.2.1). Die Lizenzen des Bluetooth Development Kit JSR-82 von Avetana GmBH, welches in dieser Arbeit benutzt wurde, ist schnittstellengebunden. Die Einzellizenz kostet 25,- Euro, wir haben uns aber für die 10er-Lizenz entschieden, um verschiedene Schnittstellen testen zu können. Die 10er-Lizenz kostete 80,- Euro.

Die Software für diese Arbeit ist in JAVA geschrieben worden, was keine weiteren Kosten verursacht hat.

Für das ***PbAE***-System ist eine Netzwerk-Infrastruktur notwendig. Diese ist aber in den meisten Fällen schon vorhanden.

Zusammengefasst sind die Kosten des ***PbAE***-Systems für die Unternehmen überschaubar. Für den Kunden fallen nur Kosten für das mobile Gerät an, die je nach Kundenwünschen variieren können. Die mobile Applikation in dieser Arbeit ist

[1] Abhängig von der Anzahl der Abfragen, die bearbeitet werden sollen, oder der betreuten Kunden

in J2ME geschrieben, das setzt besondere Anforderungen an das mobile Gerät voraus. Das Gerät muss eine Java-Laufzeitumgebung anbieten und die Bluetooth-Schnittstelle soll von JAVA steuerbar sein (JSR-82). Außerdem muss der Schreib- und Lesezugriff auf dem lokalen Dateisystem erlaubt sein (JSR-75).

2003 konnten ganz wenige Handys diese Voraussetzungen erfüllen. Aus zehn untersuchten Handys der mittleren und oberen Klasse kamen nur drei in Frage. Die Einsteigermodelle boten die JAVA-Umgebung an, jedoch in den meisten fällen keine Bluetooth-Funktion und vor allem weder JSR-75 noch JSR-82.

Mit der Zeit hat sich das Bild völlig geändert. Der größte Teil der oberen und mittleren Klasse ist für das ***PbAE***-System geeignet. Einige Einsteigermodelle mit Preisen unter 150,- Euro erfüllen auch die Anforderungen, wie das Nokia 2630 für ca. 115,- Euro (Stand August 2007).

Erfolgreich getestet ist das ***PbAE***-System mit dem Nokia N70 und dem Sony Ericsson K800i für jeweils 230,- Euro (Stand August 2007).

Auf den PDAs und MDAs sieht es leider etwas schlechter aus. Die Testgeräte Qtek 9100 und Compaq IPaq H3970 verfügten über keine MIDP 2.0 kompatible Java-Laufzeitumgebung. Eine Erweiterung mit externer Software war auch nur begrenzt möglich. Ein erfolgreiches Testen des Systems ist auf diesen Geräten nicht gelungen. Dafür wurden folgende Java Laufzeitumgebungen getestet:

**MySaifu JVM** ist eine kostenfreie „Java Virtual Machine“ für Windows Mobile. Am 16.12.2006 wurde die momentan aktuellste Version 0.3.3 veröffentlicht.

**CrE-ME** ist eine kostenpflichtige Java Laufzeitumgebung für Windows Mobile. Sie wurde von NSIcom entwickelt. Im Moment ist die Version CrE-ME 4.12 seit 01.06.2007 verfügbar.

**J9 runtime** ist Bestandteil des IBM Projekts „WebSphere Everyplace Micro Environment“ und kostenpflichtig. Die aktuellste verfügbare Version ist zurzeit die Version 6.1.1.

Tests mit den neuen PDA- bzw. MDA-Modellen sind aus zeitlichen und finanziellen Gründen nicht durchgeführt worden.

# Kapitel 11

# Zusammenfassung und Schlusswort

In dem Prospekt „Reisepass & Personalausweis“ [Bun] der Bundesdruckerei wird auf die Notwendigkeit moderner Authentifizierungssysteme hingewiesen.

> Seine persönlichen Daten und Informationen sicher zu bewahren und jederzeit verfügbar zu haben, gehört zu den Grundbedürfnissen des modernen Menschen. Reisen, internationaler Austausch und die Möglichkeit, sich überall auf der Welt sicher und frei bewegen zu können, sind Forderungen, die mit der Globalisierung auch neue technische Sicherheitslösungen im internationalen Grenzverkehr erforderlich gemacht haben.
>
> Schon längst reicht das einfache Ausweis-“Papier“ nicht mehr aus, um nationalen und internationalen Sicherheitsstandards gerecht zu werden. Zahlreiche hochkomplexe Sicherungssysteme sind notwendig, um die datentechnische Identität des Individuums zu schützen und eine verlässliche Verbindung zwischen Dokumenteninhaber und ID-Dokument herstellen zu können.

Diese Arbeit ist ein Vorschlag, wie die Authentifizierung und Verschlüsselung mit einem einzigen System durchgeführt werden können. Das System gewährleistet auch die Anonymität des Benutzers und bietet eine für viele Einsatzbereiche ausreichende Sicherheit.

## 11.1 Vor- und Nachteile

Das Bild, welches der Dienstleister bzw. der Kontrolleur zu sehen bekommt, verrät nichts weiter als das, was man ohnehin schon kennt. Der Kunde steht im Normalfall dem Kontrolleur gegenüber, deswegen ist das Bild nichts, was die Anonymität des Kunden verletzen würde.

Aus den Daten Rückschlüsse über den betroffenen Kunden zu machen, ist nicht möglich, wenn man nur über das Bild verfügt.

Das ***PbAE***-System ist ein Forschungsprojekt, das noch nicht ausgereift ist. Es ist als Grundbaustein für weitere Projekte gedacht. Erweiterungen und Verbesserungen sind notwendig, bis es ein einsatzbereites System wird.

Die für das ***PbAE***-System erforderliche Hardware ist in den meisten Fällen schon vorhanden. Die Erweiterung etwa mit einer Bluetooth-Schnittstelle ist mit geringen Kosten möglich.

Die Software-Kosten sind überschaubar. Die Entwicklung komplexerer und sicherer Applikationen ist aber mit höheren Kosten verbunden.

Einer der wichtigen Nachteile des Systems ist, dass der private Schlüssel auf dem mobilen Gerät mitgeführt wird. Das Gerät kann gestohlen werden oder verloren gehen. Der Schlüssel kann aber mit einem Passwort geschützt werden, dann ist das Gerät genauso geschützt wie alle herkömmlichen „Besitz“-Systeme.

Weitere Nachteile des ***PbAE***-Systems, die bei den konventionellen Verfahren nicht zu finden sind, sind mir nicht bekannt, ich kann sie aber nicht ausschließen.

## 11.2 Auswirkungen

Der Endverbraucher kann ein ***PbAE***-System für verschiedene Szenarien einsetzen, was z.B. mit den Kundenkarten nicht möglich ist, was ihm das Verwalten wesentlich erleichtert. Es muss, wenn überhaupt, nur noch ein Passwort gemerkt werden, mit dem das System und die Schlüssel geschützt sind.

Die Daten können auch auf einem Server gesichert werden und nicht nur auf tragbaren Medien transportiert werden, wie dies für herkömmliche Tickets, Urkunden oder Zeugnisse der Fall ist. Somit wird das Risiko, sie zu verlieren, minimiert.

Das Verschlüsseln der Daten verhindert, dass Dritte auf sie zugreifen, was die Privatsphäre des Benutzers schützt.

## 11.3 Schlusswort

Ich habe mit dieser Arbeit die grundsätzliche Machbarkeit einer identitätsbasierten Kryptographie mit einem Bild als öffentlichem Schlüssel gezeigt. Ich hoffe, dass andere Wissenschaftler diese Ideen aufgreifen und weiterentwickeln werden.

# Teil III

# Anhang

# Literaturverzeichnis

*Alle angegebenen Web-Seiten wurden vor Abgabe der Arbeit am 5. und 6. April 2008 nochmals aufgerufen und auf das verwendete Zitat hin überprüft. Auf die Angabe des Zugriffstages bei den einzelnen Verweisen wurde daher verzichtet.*


[Ali05] Ali, Saqib: *Diffie-Hellman Key Exchange.* http://www.benamar.org/url/?ali05, 2005

[Are] AreaMobile, AG: *AreaMobile - Handys, News und Testberichte.* http://www.areamobile.de

[Ave] Avetana, GmbH: *avetanaBluetooth JSR82 implementation.* http://www.avetana-gmbh.de/avetana-gmbh/produkte/Readme.xml

[Bar05] Barreto, P.-S.-L.-M.: The Pairing Based Crypto Lounge. (2005). http://www.benamar.org/url/?Bar05

[BBDD05] Burnett, Andrew ; Byrne, Fergus ; Dowling, Tom ; Duffy, Adam: *Identity Based Encryption Provider.* http://www.crypto.cs.may.ie/software/eyebee/, 2005

[BBDD06] Burnett, Andrew ; Byrne, Fergus ; Dowling, Tom ; Duffy, Adam: *A Biometric Identity Based Signature Scheme.* http://www.crypto.cs.nuim.ie/papers/ijns2006.pdf, 2006

[Beu96] Beutelspacher, Albrecht: *Kryptologie.* 5. verb. Aufl. Vieweg, 1996

[BF01] Boneh, Dan ; Franklin, Matt: Identity-Based Encryption from the Weil Pairing. In: *Lecture Notes in Computer Science* 2139 (2001), 213–229. citeseer.ist.psu.edu/boneh01identitybased.html

[Blu] BluetoothSIG, Inc.: *Bluetooth - Learn.* http://www.bluetooth.com/Bluetooth/Learn/

[BSI] BSI: *Anwendungen auf der Basis des MIX-Modells.* http://www.bsi.bund.de/literat/anonym/anwmix.htm

[BSW04] Beutelspacher, A. ; Schwenk, J. ; Wolfenstetter, K. D.: *Moderne Verfahren der Kryptographie : von RSA zu Zero-Knowledge.* 5. verb. Aufl. Vieweg, 2004

[Buc01] Buchmann, Johannes: *Einführung in die Kryptographie.* 2. erw. Aufl. Springer Verlag, 2001

[Bun] Bundesdruckerei: Reisepass & Personalausweis. http://www.bundesdruckerei.de : Bundesdruckerei GmBH. – Broschüre

[Cad03] Cadena, Robert: *KXML: A Great Find for XML Parsing in J2ME.* http://www.devx.com/xml/Article/11773/0/page/3, April 2003

[Cas] Castle, Bouncy: *Bouncy Castle Crypto APIs for Java.* http://www.bouncycastle.org

[Cau06] Caumanns, Jörg: Der Patient bleibt Herr seiner Daten. In: *Informatik Spektrum* (Mai 2006)

[CC03] Cha, J.C. ; Cheon, J.H.: An Identity-Based Signature from Gap Die-Hellman Groups. 2567 (2003), S. 18–30

[Cer07] Certicom: *Online Elliptic Curve Cryptography Tutorial.* http://www.certicom.com/index.php?action=ecc_tutorial,home, 2007

[Cha81] Chaum, David: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.* http://world.std.com/~franl/crypto/chaum-acm-1981.html, 1981

[Cha85] Chaum, David: Security without Identification: Transaction Systems to make Big Brother Obsolete. In: *Communications of the ACM* 28 (October 1985), S. 1030–1044

[Cha87] Chaum, David: Security without Identification: Card Computers to make Big Brother Obsolete. In: *DigiCash* (1987)

[Cha88] Chaum, David: *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability.* http://www.nyx.net/~awestrop/crypt/diningcr.htm, 1988

[Coc01] Cocks, Clifford: An Identity Based Encryption Scheme based on Quadratic Residues, In 8th IMA International Conference on Cryptography and coding. (2001), S. 360–363

[CS07] Chakrabarti, Saikat ; Singhal, Muskesh: Password-Based Authentication: Preventing Dictionary Attacks. In: *Computer - IEEE Computer Society* (Juni 2007)

[Dif88] Diffie, Whitfield: *The first ten years of public-key cryptography.* http://cr.yp.to/bib/1988/diffie.pdf, 1988

[DORS07] Dodis, Yevgeniy ; Ostrovsky, Rafail ; Reyzin, Leonid ; Smith, Adam: *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data.* http://www.cs.bu.edu/~reyzin/papers/fuzzy.pdf, 2007

[DR99] Daemen, Joan ; Rijmen, Vincent: *AES Proposal: Rijndael.* http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf, September 1999

[Ewa05] Ewald, Gerd: *IDEA.* http://www.regenechsen.de/phpwcms/index.php?krypto_idea, August 2005

[Fen00] Feng, Yu: *Network Programming with J2ME Wireless Devices.* http://www.benamar.org/url/?fen00, 2000

[FJP96] Federrath, Hannes ; Jerichow, Anja ; Pfitzmann, Andreas: *MIXes in mobile communication systems: Location management with privacy.* http://www-sec.uni-regensburg.de/publ/1996/FeJP_96.pdf, 1996

[Fla05] Flanagan, David: *Java in a Nutshell.* 5. Aufl. O'REILLY, März 2005

[FP98] Federrath, Hannes ; Pfitzmann, Andreas: „Neue" Anonymitätstechniken. Eine vergleichende Übersicht. In: *DuD. Datenschutz und Datensicherheit 22* (1998)

[FS87] Fiat, Amos ; Shamir, Adi: How to prove yourself: Practical solutions to identification and signature problems. (1987), 186–194. citeseer.ist.psu.edu/fiat87how.html

[FZ01] Feng, Yu ; Zhu, Jun: *Basic Network Programming in J2ME MIDP.* http://www.benamar.org/url/?fz01, 2001

[Gar95] Garfinkel, Simson: *PGP Pretty Good Privacy.* 1st Edition. O'REILLY, 1995

[GH02] Gouda, Mohamed G. ; Huang, Chin-Tser: *A Secure Address Resolution Protocol.* http://www.cs.utexas.edu/~chuang/TR-02-34.ps, Juni 2002

[Gig03] Giguere, Eric: *Using Threads in J2ME Applications.* http://developers.sun.com/mobility/midp/articles/threading2/, Februar 2003

[Gri06] Grimes, Roger A.: *MySpace password exploit: Crunching the numbers (and letters).* http://www.benamar.org/url/?gri06, November 2006

[Hei] Heise.de: *Verschiedene Artikel.* http://www.heise.de

[Hel01] Hellman, Martin E.: Die Mathematik von Public-Key-Verfahren. In: *Spektrum der Wissenschaft - Dossier Kryptographie.* (April 2001)

[Hes02] Hess, Florian: Efficient identity based signature schemes based on pairings. 2595 (2002), S. 310–324. – proceedings of SAC 2002.

[Heu] Heumann, Björn: *Biometrische Identifikation.* http://www.biometrie-online.de

[HL] Herrmann, Thomas ; Loser, Kai-Uwe: *Datenschutz.* http://www.benamar.org/url/?hl

[HMV04] Hankerson, D. ; Menezes, A. ; Vanstone, S.: *Guide to elliptic curve cryptography*. 5. verb. Aufl. Vieweg, 2004

[Hof] Hof, Thurner: *Ali Baba*.
http://de.wikipedia.org/wiki/Ali_Baba

[Hopa] Hopkins, Bruce: *Getting Started with Java and Bluetooth*.
http://today.java.net/pub/a/today/2004/07/27/bluetooth.html

[Hopb] Hopkins, Bruce: *Wireless J2ME Applications with Java and Bluetooth*.
http://www.benamar.org/url/?hopb

[Hro04a] Hromkovic, Juraj: *Theoritical Computer Science*. 1. Aufl. Springer Verlag, 2004

[Hro04b] Hromkovic, Juraj: *Theoritische Informatik*. 2. Aufl. B.G. Teubner Verlag, 2004

[IBM] IBM: *WebSphere Everyplace Micro Environment*.
http://www.ibm.com/software/wireless/weme/

[idb] *ID-based cryptography*. : *ID-based cryptography*.
http://en.wikipedia.org/wiki/ID-based_cryptography

[int] intermedia, GbR inside: *Handy Mobilfunk Portal*.
http://www.inside-handy.de

[IUS03] ICPP ; ULD ; SNG: *Identity Management Systems (IMS): Identification and Comparison Study*.
http://www.benamar.org/url/?ius03, 2003

[Jen03] Jendricke, Uwe: *Sichere Kommunikation zum Schutz der Privatsphäre durch Identitätsmanagement*. Rhombos-Verlag, 2003. – 150 S.
http://www.rhombos.de/onlinesh/jjeo/produkte/uwe.htm

[KBG] Könnecke, Christian ; Blossey, Daniel ; Gelies, Franziska: Iris Recognation, Multimedia & Security.
http://wwwiti.cs.uni-magdeburg.de : Institut für Technische und Betriebliche Informationssysteme der Universität Magdeburg. – Forschungsbericht

[Kle07] Klein, Andreas: Visuelle Kryptographie. (2007)

[KMH07] Klingsheim, Andre N. ; Moen, Vebjorn ; Hole, Kjell J.: Challeges in Securing Networked J2ME Applications. In: *Computer - IEEE Computer Society* (Februar 2007)

[LQ04] Libert, Benoît ; Quisquater, Jean-Jacques: The exact security of an identity based signature and its applications. Version: 2004. citeseer.ist.psu.edu/libert04exact.html. 2004 (Cryptology ePrint Archive 2004/102)

[Lyn06] Lynn, Ben: *PBC Library Manual*.
http://crypto.stanford.edu/pbc/manual.pdf, 2006

[Mah03a] Mahmoud, Qusay H.: *J2ME Low-Level Network Programming with MIDP 2.0.* `http://developers.sun.com/mobility/midp/articles/midp2network`, April 2003

[Mah03b] Mahmoud, Qusay H.: *The Java APIs for Bluetooth Wireless Technology.* `http://developers.sun.com/mobility/midp/articles/bluetooth2/`, April 2003

[McL00] McLaughlin, Brett: *Java and XML.* 1. Aufl. O'REILLY, 2000

[Mic] Microsystems, Sun: *Connected Limited Device Configuration .* `http://java.sun.com/products/cldc/index.jsp`

[MJ01] Markotten, Daniela G. ; Jendricke, Uwe: Identitätsmanagement im E-Commerce. In: *it+ti Informationstechnik und Technische Informatik* 43 (2001), October, Nr. 5, 236–245. `http://www.benamar.org/url/?mj01`

[MOV97] Menezes, Alfred J. ; Oorschot, Paul C. ; Vanstone, Scott A.: *Handbook of Applied Cryptography.* CRC Press, 1997

[Mys] Mysaifu: *Mysaifu JVM.* `http://www.benamar.org/url/?mys`

[NSI] NSIcom: *NSIcom CrE-ME.* `http://www.nsicom.com/`

[Pat02] Paterson, Kenneth: ID-based signatures from pairings on elliptic curves. (2002). `citeseer.ist.psu.edu/paterson02idbased.html`

[PB07] Pape, Sebastian ; Benamar, Nabil: Using Identity-Based Public-Key Cryptography with Images to Preserve Privacy. In: Pre-Proceedings of the Third IFIP / FIDIS Summer School, 'The Future of Identity in the Information Society', Karlstad, August 2007 / Praktische Informatik, Kassel. 2007. – Forschungsbericht

[PM04] Pashalidis, Andreas ; Mitchell, Chris J.: *A Security Model for Anonymous Credential Systems.* `http://www.isg.rhul.ac.uk/~xrtc/cv/credModel.pdf`, 2004

[Pro] Process, Java C.: *JSR 82: Java APIs for Bluetooth.* `http://www.jcp.org/en/jsr/detail?id=82`

[QG88] Quisquater, Jean-Jacques ; Guillou, L.: A 'Paradoxical' Identity-Based Signature Scheme Resulting from Zero-Knowledge. 0403 (1988), S. 216–231

[Ras05] Rash, Wayne: *TriCipher Ships Multipart Authentication System.* `http://www.eweek.com/article2/0,1895,1764227,00.asp`, Februar 2005

[RLS99] Rawolle, J. ; Lassahn, C. ; Schumann, M.: Wege zur Absicherung eines Intranets. In: *Informatik Spektrum 22* (1999)

[Roß96] Rossnagel, Alexander: *Verordnung zur digitalen Signatur.* `http://www.benamar.org/url/?Ross96`, 1996

[Sch] Schall, Christoph: Bluetooth API (JSR-82). `http://www.benamar.org/url/?sch` : Universität Ulm. – Proseminar

[Sch06a] Schneier, Bruce: *MySpace Passwords Aren't So Dumb.* `http://www.schneier.com/essay-144.html`, Dezember 2006

[Sch06b] Schneier, Bruce: *Schneier on Security: Real-World Passwords.* `http://www.benamar.org/url/?sch06b`, Dezember 2006

[Sch07a] Schmidt, Jürgen: *Online Banking fatal.* `http://www.heise.de/security/artikel/94451`, August 2007

[Sch07b] Schmidt, Jürgen: *Passwortklau für Dummies.* `http://www.heise.de/security/artikel/94100`, August 2007

[Sch07c] Schneier, Bruce: *Secure Passwords Keep You Safer.* `http://www.schneier.com/essay-148.html`, Januar 2007

[Sec] Seceidos: *Two-Factor-Authentication.* `http://www.seceidos.de/de/service/glossar/`

[Sei] Seibold, Michael: *Cipherbox - Die Krypto-Homepage, eine Einfuehrung in die Kryptologie, Datensicherheit und verschiedene Verschluesselungsverfahren, mit Software-Download.* `http://www.cipherbox.de`

[Sha85] Shamir, Adi: Identity-based cryptosystems and signature schemes. Proceedings of CRYPTO 84 on Advances in cryptology. (1985), S. 47–53

[SOK00] Sakai ; Ohgishi ; Kasahara: Cryptosystems based on pairing. (Jan. 2000). – In the Symposium on Cryptography and Information Security-SCIS'2000

[Tho07] Thomas, Norman: Realisierung einer mobilen Authentifizierung, mittels ereignis-synchroner Token / Praktische Informatik, Kassel. Januar 2007. – Diplomarbeit

[Top02] Topley, Kim: *J2ME in a Nutshell.* 1. Aufl. O'REILLY, 2002

[Vera] VeriSign: *Unified Authentication Tokens.* `http://www.verisign.de/unified-authentication/usb-tokens/`

[Verb] Verschiedene: *Verschiedene Artikel.* `http://de.wikipedia.org`

[Ver06] Verschiedene: *FuE-Projekt: Die Spezifikation der elektronischen Gesundheitskarte (Teil 1, 2 und 3).* `http://www.dimdi.de/`, 2006

[wika] WIKIBOOKS.ORG: *RFID-Technologie*.
http://de.wikibooks.org/wiki/RFID-Technologie

[Wikb] WIKIPEDIA: *Transaktionsnummer*.
http://de.wikipedia.org/wiki/Transaktionsnummer