DIPLOMARBEIT

Cand. -Math. : Dac-Nhon, Ho
Matr. Nr. : 583569

Schlüsselregistrierung und -rückgewinnung mit redundanter und verteilter Kontrolle

Technische Universität Darmstadt
Fachbereich Mathematik

angefertigt im
Fachbereich Informatik
Fachgebiet Graphisch-Interaktive Systeme

Betreuer : Dipl.-Inform. Volker Roth
Prüfer : Prof. Dr.-Ing. José L. Encarnação

Eidesstattliche Erklärung

Hiermit versichere ich an an Eides Statt, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen verwendet habe.

Darmstadt, den 02.06.1998

Dac Nhon, Ho

Danksagung

Für die Geduld meines Betreuers, Herrn Dipl.-Inform. Volker Roth, danke ich ihm herzlich. Ohne seinen persönlichen Einsatz konnte ich die Diplomarbeit aus organisatorischen Gründen nicht fortsetzen.

Inhaltsverzeichnis

1 Einführung **9**

1.1 Motivation und Problemstellung 9

1.2 Entwicklung von Secret Sharing Schemes 9

1.3 Ziel der Arbeit 11

1.4 Organisation der Arbeit 11

2 Grundlagen **13**

2.1 Zahlentheorie endlicher Körper 13

2.2 Grundlagen der Wahrscheinlichkeits- und Informationstheorie 15

2.2.1 Grundbegriffe aus der Wahrscheinlichkeitstheorie 16

2.2.2 Grundbegriffe aus der Wahrscheinlichkeitstheorie 16

2.2.3 Grundbegriffe der Informationstheorie 21

2.2.4 Grundbegriffe aus der Informationstheorie 27

2.3 Komplexität 27

2.4 Kryptographische Grundlagen 28

2.4.1 Kryptosysteme 28

2.4.2 Sicherheit von Kryptosystemen 29

2.4.2.1 Uneingeschänkte Sicheheit (*unconditional security*) . 30

2.4.2.2 Berechenbare und beweisbare Sicheheit (*computational security*) 30

2.4.3 Kryptoanalyse 31

2.4.3.1 Angriffe auf die kryptographischen Verfahren 32

2.4.3.2 Angriff auf kryptographische Protokolle 34

2.4.4 kryptographische Primitiven 35

2.4.4.1 krytographische Einweg-Hashfunktion 35

2.4.4.2 Signatur-Verfahren 37

2.4.4.3 Authentisierung im öffentlichen Netzwerk 42

3 Zugriffsstrukturen **45**

3.1 Terminologie 45

3.2 Threshold-Zugriffsstruktur 47

3.3 Multilevel-Zugriffsstrukturen 48

3.4 Compartmented-Zugriffsstruktur 49

4 Secret Sharing Schemes **51**

4.1 Theoretische Grundlagen 51

4.1.1 Perfekte und ideale Secret Sharing Schemes 53

4.1.2 Informationsrate (Information rate) eines PSS 56

4.2 Implementierung von Secret Sharing Schemes 58

4.2.1 Das (k, n)-Threshold-Schema (A. Shamir) 58

4.3 Effiziente Konstruktion von Secret Sharing Schemes 62

4.4 Anwendungsorientierte Secret Sharing Schemes 69

4.4.1 Das Secret Sharing Schema von Tompa und Woll 69

4.4.2 Das Verfahren von Lin und Harn 75

4.4.3 Verifiable Secret Sharing Schemes 82

5 Anwendung **91**

5.1 Anwendungsprobleme 92

5.2 Das Modell und Einschränkungen 95

5.2.1 Das Modell 95

5.2.2 Einschänkungen 98

5.3 Modifikation 100

5.4 Infrastruktur 101

5.5 Konzeption und Realisierung des Schlüsselspeichers 102

5.5.1 Bechreibung des Schlüsselspeichers 102

5.5.2 Sicherheit des Schlüsselspeichers 103

5.5.3 Speicherung und Zugriffsmechanismen 105

5.5.4 Redundante Rekonstruktion 112

5.5.4.1 Erneuerung der Shares 113

5.5.5 Authentisierung im SRS 116

5.5.6 Redundanz-Verlust verhindern 118
5.6 Das erste Konzept des Relaisierens eines SRS 118
5.7 Das zweite Konzept für die Realisierung eines SRS 120
5.7.1 Datenmodellierung im Recovery-System 121
5.7.2 Modellieren der Beziehungen 122
5.7.3 Datenbeispiel 123
5.7.4 Dienste im SRS 124
5.8 Das gesamte Konzept für die Realisierung eines SRS 126
5.9 Sicherheitspolitik nach der Rekonstruktion 127

6 Zusammenfassung und zukunftige Arbeiten 129

Kapitel 1

Einführung

1.1 Motivation und Problemstellung

Otto Horak schrieb 1994 „*Der Schutz sensibler Information ist ein Anliegen, das bis in die Anfänge menschlicher Kultur reicht*“.

Die Kryptographie löst dieses Problem durch moderne Verschlüsselungstechniken, welche auf dem Prinzip des niederländischen Philologen Kerckhoff basiert. Dieses Prinzip besagt, daß die Sicherheit eines Kryptosystems nicht von der Geheimhaltung der Ver- und Entschlüsselungsfunktion abhängt, sondern ausschließlich von der Geheimhaltung des verwendeten Schlüssels.
Was geschieht, wenn der Schlüssel nicht mehr verfügbar ist, z.B. wenn der Besitzer des Schlüssels verunglückt ist oder seine Anstellung wechselt?

Traditionelle Schlüsselverwaltung löst dieses Problem, indem kryptographische Schlüssel passiv gespeichert werden. Computersystem, in dem die Schlüsselgespeichert sind, können zum beispiel durch Hardware-Fehler ausgefallen sein, so daß die dort gespeicherten Schlüssel verloren gehen. Aus diesem Gefahr werden die Schlüssel mehrfach kopiert. Aber mehrfache Kopie schwächt die Sicherheit der Geheimhaltung der gespeicherten Schlüssel ab, denn Unbefugten können leichter an die Schlüssel dran kommen.

Für die Verwaltung besonders wichtiger Schlüssel ist *Secret Scharing Schema* geeignet. Dabei handet es sich um ein Verfahren, mit dem der Schlüssel in verschiedene Schlüsselkomonenten zerlegt und dann an die verschiedene vertrauenswürdige Personen verteilt. Der Schlüssel kann später nur durch bestimmte Konstellationen dieser Personen rekonstruiert werden, auch wenn, verfahren abhängig, einige von ihnen ausgefallen sind.

1.2 Entwicklung von Secret Sharing Schemes

In einem Secret Sharing Schema hat der *Dealer* ein Geheimnis (*Key*). Es gibt eine endliche Menge $\mathcal{W}$ von Personen, bezeichnet als Wächter, und eine Menge $\mathcal{A} \subseteq 2^{\mathcal{W}}$

von autorisierten Teilmengen von $\mathcal{W}$. Ein Secret Sharing Schema mit $\mathcal{A}$ als *Zugriffsstruktur* besteht aus zwei Prozessen: *Verteilung-* und *Rekonstruktionsprozess*. Bei dem Verteilungsprozess berechnet der Dealer aus dem Geheimnis die Teilgeheimnisse (*Shares*) und verteilt sie an jeden Wächter. Der Share ist die geheime Teilinformation des jeweiligen Wächters. In dem Rekonstruktionsprozess kann das Geheimnis nur durch eine autorisierte Teilmenge von Wächtern rekonstruiert werden. Die Sicherheit des Secret Sharing Schemas beruht darauf, daß Teilmengen, welche nicht in $\mathcal{A}$ enthalten sind (*nicht-autorisierte Teilmengen*), keine *Shannon'sche Information* über das Geheimnis erhalten. Secret Sharing Schemata, deren Sicherheit diese Eigenschaft (*unconditional security*) hat, heißen *perfekt*. Ein perfektes Secret Sharing Schema heißt *ideal*, wenn die Menge aller möglichen Werte des Geheimnisses und die Menge aller in dem Schema verwendeten Shares gleichmächtig sind. *Verallgemeinerte Secret Sharing Schemata* können beliebige Zugriffsstrukturen realisieren. Basiert die Sicherheit eines Secret Sharing Schemas auf *unbewiesene kryptographische Annahmen*, so erhalten nicht-autorisierte Teilmengen keine *berechnungssicheren (computationale) Informationen* über das Geheimnis. Mit anderen Worten, solche Secret Sharing Schemata sind *berechnungssicher*.

Eine Idee, wie man ein Geheimnis aus seineen Teilinformationen schützen kann, wurde 1979 von D. Chaum veröffentlicht [6]. Chaum nannte diese Methode als *Partial Key Technik*. Etwas später im gleichen Jahr gaben G.R. Blakley und A. Shamir unabhängig voneinander weitere Methoden für die Rekonstruktion des Geheimnisses bekannt. Die *Key safeguarding Technik* von Blakley [2] basiert auf die projektive Geometrie. Das (k, n)-*Threshold Schema* von Shamir [25] rekonstruiert das geteilte Geheimnis mit Hilfe von Lagrange'cher Interpolationsformel im $\mathbb{Z}_p$, p ist eine Primzahl.

Die Hauptidee dieser drei Methoden erlaubt, Geheimnis in eine Folge von n *Stücken* zu kodieren, so daß k oder mehr Stücke das Geheimnis rekonstruieren können. Weniger als k Stücke geben jedoch keinen Anhaltspunkt über das Geheimnis.

Obwohl diese Verfahren einfach zu realisieren sind, haben sie einige Schwachpunkte und sind zu speziell. Betrachten wir zum Beispiel das (k, n)-Threshold Schema, bei dem das Geheimnis unter n Wächtern geteilt wird. k Personen geben ihre Shares frei, um das Geheimnis zu rekonstruieren. $(k \Leftrightarrow 1)$ Wächter erhalten mit ihren Shares keine Informationen über das Geheimnis. Die Teilmenge von Wächtern, die das Geheimnis rekonstruieren kann, hat genau k Elemente. Und es gibt genau $\binom{n}{k}$ solche Teilmengen.

Verallgemeinerte Secret Sharing Schemata beschäftigen sich mit den Fragen, wer das Geheimnis rekonstruieren darf und wie wird das Geheimnis rekonstruiert. Die Antwort auf die erste Frage führt zu eine sog. Zugriffstruktur, d.h. eine Familie von Teilmengen von Wächtern, autorisierten Teilmengen. Die zweite gibt an, wie man anhand einer vorgegebenen Zugriffsstruktur ein Rechenschema entwerfen kann, das diese Zugriffsstruktur realisiert.

1.3 Ziel der Arbeit

In der Diplomarbeit wird zunächst in die theoretische Grundlagen von Secret Sharing Schemata eingeführt. Dabei werden einige Zugriffstrukturen und deren Realisierung durch verschiedene Konstruktionsmethoden vorgestellt. Danach werden die typische Secret Sharing Schemata in Bezug auf Effizient, Betrugsversuchen und Überprüfung der Shares ausführlich beschrieben.

Wichtiger Bestandteil dieser Arbeit ist ein Konzept für die Behandlung der Speicherungs- und Verwaltungsprobleme von traditionellen Secret Sharing Schemata zu geben, die durch lokale Speicherung auf jeden Wächter und ständige überprüfungen auf Korrektheit und Konsistenz der Shares Kommunikationsenpaß im Netzwerk verursachen können.
Durch die Einführung des Schlüsselspeichers können diese Probleme gelöst werden . Mit dem Schlüsselspeicher als Objekt für die Zentralspeicherung der Shares werden eingefügte Schlüssel mit hoher Redundanz verteilt zurückgewonnen.

1.4 Organisation der Arbeit

Im **Kapitel 2** Wichtige Grundlagen für die Darstellung und Beschreibung der Sachverhalten in Kapitel 3, 4 und 5.
Für die Arbeit benutzte Tatsachen aus der Zahlenstherie endlicher Körper, Wahrscheinlichkeits- und Informationstheorie sowie wichtige kryptographische Primitiven werden eingegangen

Im **Kapitel 3** werden einige Zugriffsstrukturen definiert. Durch die Zugriffsstrukturen wird in diesem Kapitel erklärt, wer die Schlüssel rekonstruieren darf und wer nicht. Als spezielle Zugriffsstrukture wird die Threshold-Zugriffsstrukture behandelt. Allgemeine Zugriffsstrukturen wird zu monotonen gefaßt. Darunter werden Multilevel- und Compartmented-Zugriffsstruktur formal beschrieben.

Das **Kapitel 4** wird in zwei Teile gegliedert. Im ersten Teil werden theoretische Grundlagen von Secret Sharing Schemata sowie Ergebnisse für effiziente Realisierung von Secret Sharing Schemata zusammengefaßt. Der zweite Teil beschreibt typische Secret Sharing Schemata, welche praktische Probleme bei der Geheimhaltung bzw. Rückgewinnung des Geheimnis durch verteilte Kontrolle lösen.

Im **Kapitel 5** Werden die Nachteile typischer Speicherungsprobleme im traditionellen Secret Sharing Schemata heraus diskutiert. Diese Nachteile werden beseitigt durch die Einführung des Schlüsselspeichers. Mit diesem Hilfsmittel wird ein Konzept für besseren Schutz kryptographischer Schlüssel durch deren Verteilung. Weiter werden in diesem Kapitel Lösungsansätze für die Speicherungs-, Verwaltung der Shares, so daß die registrieten Schlüssel mit hoher Redundanz verteilt rekonstruiert werden können.

Zum Schluß wird im **Kapitel 6** zusammengefaßt, welche Probleme in dieser Arbeit gelöst wurden und welche noch zu lösen sind.

Kapitel 2

Grundlagen

2.1 Zahlentheorie endlicher Körper

Satz 2.1. Es seien p und q zwei Primzahlen mit der Eigenschaft $q|(p \Leftrightarrow 1)$. $\mathbb{Z}_p$ und $\mathbb{Z}_q$ seien die zugehörige Restklassenkörper.
Dann existieren genau $(q \Leftrightarrow 1)$ Zahlen $g_i \in \mathbb{Z}_p$ $(i = 1, \ldots, q \Leftrightarrow 1)$, die je eine zyklische multiplikative Unterruppe G_i von $\mathbb{Z}_p$ der Ordnung q generiert. Diese Elemente sind $\left\{g_i^0, g_i^1, \ldots, g_i^{q-1}\right\}$. g_i heist das primitive Element der Gruppe G_i (oder der Generator, das erzeugende Element).

Satz 2.2. Es seien p und q zwei Primzahlen mit der Eigenschaft $q|(p \Leftrightarrow 1)$. $\mathbb{Z}_p$ $\mathbb{Z}_q$ seien die Restklassenkörper, welche jeweils durch p und q bestimmt sind. ferner sei $g \in \mathbb{Z}_p$ gegeben, welche eine zyklische Untergruppe von $\mathbb{Z}_p$ mit q Elementen $g^0, g^1, \ldots, g^{q-1}$ generiert. Dann gilt:

1. $g^q = 1$
2. $q|x \Leftrightarrow g^x = 1$
3. $a_1 + \cdots + a_k \mod q = b \Rightarrow g^{a_1+\cdots+a_k} \mod p = g^b \mod p$
4. $a_1 \cdot \ldots \cdot a_k \mod q = b \Rightarrow g^{a_1 \cdot \ldots \cdot a_k} \mod p = g^b \mod p$

Beweis. Zu 1.
In einer Gruppe muß das Ergebnis einer Multiplikation von mehreren Gruppenelementen und damit auch das Ergebnis einer Exponentiation wieder ein Element der Gruppe sein. Da die Untergruppe G nur q verschiedene Elementen $g^0, g^1, \ldots, g^{q-1}$ enthält, muß $g^q = g^j$ für ein j mit $0 \leq j \leq (q \Leftrightarrow 1)$ gelten. Für $j > 0$ müßte dann weiterhin $g^{q-1} = g^{j-1}$ gelten, was einen Wiederspruch zur Unterschiedlichkeit der Elemente von G darstellt. Damit folgt $j = 0$ und somit $g^q = 1$.

der erweiterte Euklidische Algorithmus

Der Erweiterte Euklidische Algorithmus wird dazu benutzt, den $t = ggT(t_1, t_2)$ in dem

Schema von Lin und Hahn [12] und außerdem zwei Zahlen a_1 und a_2 mit $a_1t_1+a_2t_2 = t$ zu berechnen.
Algorithmus

- Eingabe: t_1, t_2 mit $t_1 > t_2$.
- Ausgabe: (t, a_1, a_2) mit $t = ggT(t_1, t_2)$ und $a_1t_1 + a_2t_2 = t$.

$ggT_{ext}(t_1, t_2)$:
1. **if** $t_2 = 0$ **then return** $(t_1, 1, 0)$
2. **else** $(t', a'_1, a'_2) := ggT_{ext}(t_2, t_1 \mod t_2)$
3. $(t, a_1, a_2) := (t', a'_2, a'_1 \Leftrightarrow (t_1/t_2)a'_2)$
4. **return** (t, a_1, a_2) **fi**

Der ggT von t_2 und $t_1 \mod t_2$ wird in Zeile 3 über die Variable t' an die Ergebnisvariable t übergeben, so daß der Ausgabewert von t korrekt ist. Der rekursive Aufruf in Zeile 2. gibt ein korrektes Tripel (t', a'_1, a'_2) zurück, für dessen Werte $t' = a'_1t_2 + a'_2(t_1 \mod t_2)$ gilt. Dann kann man $(t_1 \mod t_2)$ durch $(t_1 \Leftrightarrow qt_2)$ mit $q = (t_1/t_2)$ ersetzen, also:

$$t' = a'_1t_t + a'_2(t_1 \Leftrightarrow (t_1/t_2)t_2) = a'_2t_1 + (a'_1 \Leftrightarrow a'_2(t_1/t_2))t_2.$$

Damit erhält man mit der Auswahl von $a_1 = a'_2$ und $a_2 = a'_1 \Leftrightarrow (t_1/t_2)a'_2$ in der Zeile 3. die korrekten Ergebniswerte für a_1 und a_2.

Grundlage für den **erweiterterten Euklidische Algorithmus** ggT_{ext} und seine Arbeitweise werden im folgenden beschrieben.

Mit dem erweiterterten Euklidische Algorithmus wird der größte gemeinsame Teiler t von zwei ganzen Zahlen t_1, t_2 berechnet und die zwei Zahlen a_1 und a_2 deart bestimmt, daß nach dem Lemma von *Bézout* $a_1t_1 + a_2t_2 = t$ gilt.

Lemma 2.1. *Bézout.* Sei $t = \mathrm{ggT}(t_1, t_2)$. Dann existieren zwei ganze Zahlen a_1, a_2 mit $a_1t_1 + a_2t_2 = t$

Lemma 2.2. a, b seien zwei ganze Zahlen. Dann gilt: $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a \mod b))$.

Für einen Aufruf $t = \mathrm{ggT}_{ext}(t_1, 0)$ liefert der Algorithmus in der 1-ten zeile $(t, a_1, a_2) = (t_1, 1, 0)$ als korrektes Ergebnis zurück, da für diese Werte gilt:

$$t = t_1 = t = \mathrm{ggT}(t_1, 0) \quad \text{und} \quad a_1t_1 + a_2t_2 = 1t_1 + 0t_2 = t$$

Der rekursive Aufruf von ggT in der 2-ten Zeile als Ergebnis ein korrektes Tripel (t', a'_1, a'_2) zurück gibt.
Für einen Aufruf $\mathrm{ggT}_{ext}(t_1, t_2)$ wird $\mathrm{ggT}(t_1, t_2)$ mit dem rekursiven Aufruf $\mathrm{ggT}_{ext}(t_2, t_1 \mod t_2)$ in der 2-ten Zeile bestimmt. Nach dem Lemma gilt $\mathrm{ggT}(t_1, t_2) = \mathrm{ggT}(t_2, t_1 \mod t_2)$.
Der größte gemeinsame Teiler $\mathrm{ggT}(t_2, t_1 \mod t_2)$ wird in der 3-ten Zeile über die Variable t' an die Variable t übergeben, so daß der Ausgabewert von t korrekt ist.

Der rekursive Aufruf in der 2-ten Zeile gibt ein korrektes Tripel (t', a'_1, a'_2) mit $t' = a'_1 t_2 + a'_2(t_1 \mod t_2)$ zurück. Dann kann man $(t_1 \mod t_2)$ durch $t_1 \Leftrightarrow qqt_2$ mit $q := \frac{t_1}{t_2}$ ersetzen, so daß

$$t' = a'_1 t_2 + a'_2(t_1 \Leftrightarrow (\frac{t_1}{t_2})t_2) = a'_2 t_1 + (a'_1 \Leftrightarrow a'_2(\frac{t_1}{t_2}))t_2.$$

Damit bekoomt man durch Wahl von $a_1 = a'_2 \quad$ und $\quad a_2 = a'_1 \Leftrightarrow (\frac{t_1}{t_2})a'_2$ in der 3-ten Zeile die korrekte Ergebniswerte für $a_1\, a_2$.

Der erweiterte Algorithmus kann nun verwendet werden, um den größten gemeinsamen Teiler t der Zahlen $t_1, \ldots, t_k$ und die entsprechende $a_1, \ldots, a_k$ mit $\sum_{i=1}^{k} a_i t_i = t$ zu bestimmen. Es gilt somit:

$$\mathrm{ggT}(t_1, \ldots, t_k) = \mathrm{ggT}(\ldots(\mathrm{ggT}(\mathrm{ggT}(t_1, t_2), t_3), \ldots, t_k).)$$

Der $\mathrm{ggT}(t_1, \ldots, t_k)$ kann daher mit einer $(k \Leftrightarrow 1)$-fach rekursiven Anwendung des erweiterten Euklidischen Algorithmus berechnet werden, wobei der $\mathrm{ggT}(t_1, \ldots, t_{m-1})$ mit t_m als Eingabe für den Algorithmus für die Bestimmung von $\mathrm{ggT}(t_1, \ldots, t_m)$ ist. Damit erhält man:

$\mathrm{ggT}(t_1, t_2) = t_{h,1}$ und $a_{1,1}, a_{1,2}$ mit $a_{1,1}t_1 + a_{1,2}t_2 = t_{h,1}$
$\mathrm{ggT}(t_{h,1}, t_3) = t_{h,2}$ und $a_{2,1}, a_{2,2}$ mit $a_{2,1}t_{h,1} + a_{2,2}t_3 = t_{h,2}$ und
$\vdots$
$\mathrm{ggT}(t_{h,(k-2)}, t_k) = t_{h,k}$ und $a_{(k-1),1}, a_{(k-1),2}$ mit $a_{(k-1),1}t_{h,(k-2)} + a_{(k-1),2}t_k = t$.

Durch Rückwärtiges Einsetzen folgt:

$$t = a_{(k-1),1}(a_{(k-2),1}(\ldots(a_{1,1}t_1 + a_{1,2}t_2)\ldots)a_{(k-2),2}t_{k-1}) + a_{(k-1),2}t_k.$$

Aus dieser Formel werden $a_1, \ldots, a_k$ mit $a_1 t_1 + \cdots a_k t_k = t$ wie folgt berechnet:

$$\begin{aligned}
a_1 &= a_{(k-1),1}a_{(k-2),1}\cdots a_{2,1}a_{1,1}\\
a_2 &= a_{(k-1),1}a_{(k-2),1}\cdots a_{2,1}a_{1,2}\\
a_3 &= a_{(k-1),1}a_{(k-2),1}\cdots a_{3,1}a_{2,2}\\
&\vdots\\
a_i &= a_{(k-1),1}a_{(k-2),1}\cdots a_{i,1}a_{(i-1),2}\\
&\vdots\\
a_k &= a_{(k-1),2}
\end{aligned}$$

2.2 Grundlagen der Wahrscheinlichkeits- und Informationstheorie

Wie in jeder kryptographische Anwendung spielt die Sicherheit eines kryptographischen Verfahren eine wichtige Rolle. Mit Hilfe von Wahrscheinlichkeitstheorie kann

man die Wahrscheinlichkeit angeben, wie Betrüger erkannt werden können. Um die Sicherheit eines Verfahren zu gewährleisten, muß die sog. *Betrugwahrscheinlichkeit* stets vorher festgelegt werden.
In jedem Secret Sharing Schema ist die Anzahl der Shares, die von einer verantwortlichen Instanz (dealer oder Trusted Key Center) erzeugt und anschließend an die Beteiligen verteilt, eine wichte Größe. Die Sicheheit eines Secret Sharing Schemas hängt von dieser Größe ab. Denn die Shares sind geheime Teilinformationen, mit denen autorisierte Teilnehmer das Geheimnis rekonstruieren können. Mit der Informationstheorie, insbesondere mit der Entropie, wird diese Größe analysiert. Dabei versucht man eine untere Schranke für die Anzahl der Shares in einem *perfekten* Secret Sharing Schema herzuleiten.

2.2.1 Grundbegriffe aus der Wahrscheinlichkeitstheorie

2.2.2 Grundbegriffe aus der Wahrscheinlichkeitstheorie

Bei der Durchführung von Experimenten kann eines von mehreren möglichen Ergebnissen auftreten. Diese Versuchergebnisse bezeichnet man als *Ereignisse*. Experimente, die durch Zufall bedingt sind, z.B. das Werfen einer Münze, nennt man *Zufallexperimente*. Bei einem Zufallexperiment können neue Ereignisse aus anderen Ereignissen durch bestimmte Verküpfungen gewonnen werden. Folgende Bezeichnungen sind in der Wahrscheinlichkeitstheorie geläufig:

- Das Ereignis $A_1 \cap A_2 = A_1 A_2$ nennt man den Durchschnitt oder das Produkt von A_1 und A_2. Das Ereignis $A_1 A_2$ tritt genau dann ein, wenn sowohl A_1 als auch A_2 eintreten, wenn also A_1 und A_2 gleichzeitig eintreten.

- Das Ereignis $A_1 \cup A_2 = A_1 + A_2$ bezeichnet man als die Vereinigung oder Summe von A_1 und A_2. Das Ereignis $A_1 + A_2$ tritt genau dann ein, wenn A_1 oder A_2 oder beide eintreten, wenn also mindesten ein von beiden eintritt.

- $\overline{A}$ heißt *Komplementär* von A. $\overline{A}$ tritt genau dann ein, wenn A nicht eintritt.

- $A_1 \setminus A_2$ heißt *Differenz* von A und B. $A_1 \setminus A_2$ tritt genau dann ein, wenn A_1 eintritt und A_2 nicht.

- Zwei Ereignisse A_1 und A_2 heißt *disjunkt*, wenn sie nicht gleichzeitig eintreten können, d. h. $A_1 \cap A_2 = \varnothing$. In diesem Sprachgebrauch heißt $\varnothing$ das *unmögliche Ereignis*. Das Ereignis, das bei jeder Durchführung eines Experiments eintritt, heißt das *sichere Ereignis*.

- Ein Ereignis, welches von $\varnothing$ verschieden ist und nicht als Summe von zwei Ereignissen darstellen läßt, heißt *Elementarereignis*.

Als Maß fur die Beschreibung des Eintreten eines Ereignisses A dient der Ausdruck $p(A)$. Er gibt an, mit welcher Wahrscheinlichkeit das Ereignis A bei einem Zufallex-

periment eintritt. Zunächst beschreiben wir den Begriff *Wahrscheinlichkeit* axiomatisch und beschränken uns dabei wegen Zweckmäsigkeit auf endliche Wahrscheinlichkeitsräume.

Definition 2.1. Es seien $A = \{a_1, \ldots, a_n\}$ die Menge der Elementarereignisse, $\mathcal{P}(A)$ die Potenzmenge von A und p ein Wahrscheinlichkeitsmaß, das jedem Ereignis $U \in \mathcal{P}(A)$ ein $p(U)$ zuordnet. Dann heißt der Tripel $(A, \mathcal{P}(A), p)$ ein endlicher Wahrscheinlichkeitsraum. In einem Wahrscheinlichkeitsraum gelten die folgenden Eigenschaften (Axiomen nach *Kolmogoroff*):

- $\forall U \in \mathcal{P}(A): \quad p(U) \geq 0$
- $p(A) = 1$
- $\forall\, U, V \in \mathcal{P}(A)$ mit $U \cap V = \varnothing: \quad p(U \cup V) = p(U) + p(V).$
- Für Elementarereignisse a_i mit $p(a_i) := p_i$ gilt: $\sum_{i=1}^{n} p_i = 1.$

Aus dieser Definition werden einige Regeln hergeleitet, mit denen die Berechnung von Wahrscheinlichkeiten vereinfacht wird.

- Für jedes Ereignis A gilt: $p(A) = 1 \Leftrightarrow p(\overline{A})$.
- Das unmögliche Ereignis $\varnothing$ hat die Wahrscheinlichkeit Null.
- Aus $A \subset B$ folgt $p(A) \leq p(B)$. Dabei bedeutet $A \subset B$: Aus dem Eintreten von A folgt das von B.
- Für beliebige Ereignisse A und B gilt: $p(B\ A) = p(B\overline{A}) = p(B) \Leftrightarrow p(BA)$.

Laplace-Experiment

Wir betrachten nun Zufallexperimente, bei denen die Menge der Elementarereignisse (a) *endlich* ist und ihre Elemente als paarweise verschiedene Versuchergebnisse (b) *gleiche Wahrscheinlichkeit* haben. Zufallexperimente mit diesen beiden Bedingungen nennt man *Laplace-Experimente*.
Es sei $A = \{a_1, \ldots, a_n\}$ Menge der Elementarereignisse mit $p(a_1) = \cdots p(a_n) = p$. Dann gilt:

$$1 = p(A) = \sum_{i=1}^{n} p(a_i) = n \cdot p \Leftrightarrow p = \frac{1}{n}$$

Ist $U = \{a_{i_1}, \ldots, a_{i_k}\}$, $1 \leq i, k \leq n$, ein Ereignis, welches aus k verschiedenen Versuchergebnissen besteht, so besitzt U die Wahrscheinlichkeit

$$p(U) = \sum_{j=1}^{k} p(a_{i_j}) = k \cdot p = \frac{k}{n} = \frac{|U|}{|A|},$$

wobei k die Anzahl der für U güntigen Fälle und n die der insgesamt möglichen Fälle ist.
Bei vielen Experimenten kann man davon ausgehen, daß es sich um Laplace-Experiment handelt. Obwohl die Menge A abzählbar endlich viele Elemente besitzt, kann man ihre Mächtigkeit als die Anzahl der insgesamt möglichec Fälle nicht immer angeben. Mit Hilfe von kombinatorischen Methoden läßt sich die Machtigkeit und damit die Wahrscheinlichkeit bei Laplace-Experimenten berechnen. Dazu sind folgenden Fakten aus der Kombinatorik nützlich.

- *Anordnungsmöglickeiten von n Elementen*

  n verschiedene Elemente können unter Berücksichtigung der Reihenfolge auf $n!$ Arten angeordnet.
  n Elemente, von denen jeweils $n_1, \ldots, n_k$ mit $n_1 + \cdots + n_k = n$ identisch sind, können auf $\dfrac{n!}{n_1! \cdot n_k!}$ verschiedene Arten angeordnet werden. Ist $n_1 = k$ und $n_2 = n \Leftrightarrow k$, so ist die Anzahl der verschiedenen Anordnungen ist $\binom{n}{k}$.

- *Auswahlsmöglickeiten aus n Elementen*

  Aus n verschiedenen Elementen kann man unter Berücksichtigung der Reihenfolge k Elemente $(1 \leq k \leq n)$ auf $n(n \Leftrightarrow 1)(n \Leftrightarrow 2) \cdots (n \Leftrightarrow k+1)$ verschiedene Arten auswählen.
  Aus n verschiedenen Elementen kann man ohne Berücksichtigung der Reihenfolge k Elemente $(1 \leq k \leq n)$ auf $\binom{n}{k}$ verschiedene Arten auswählen.

Die von Wahrscheinlichkeiten gemaß dieser kombinatorischen Methoden widerspiegeln sich bei den *Urnenmodellen*, auf die wir nicht eingehen möchten.

Bedingte Wahrscheinlichkeit

Definition 2.2. A und B seien zwei beliebige Ereignisse mit $p(B) > 0$. Dann heißt die durch

$$p(A \mid B) = \frac{p(AB)}{p(B)} \tag{2.1}$$

definierte Zahl $p(A \mid B)$ *bedingte Wahrscheinlichkeit* von A unter Bedingung B.

Das ist die Wahrscheinlichkeit, mit der das Ereignis A eintritt wenn bekannt ist, daß das Ereignis B eingetreten ist. Direkt aus dieser Definition folgt der *Multiplikationssatz.*

Satz 2.3. Für zwei beliebige Ereignisse A, B gilt:

$$p(AB) = p(A \mid B) \cdot p(B) \tag{2.2}$$

Durch vollständige Induktion gilt der Satz auch für n Ereignisse. Es seien $A_1, \ldots, A_n$ beliebige Ereignisse, dann gilt nach dem *Multiplikationssatz*:

$$\begin{aligned} p(A_n A_{n-1} \ldots A_1) \;=\; & p(A_n \mid A_{n-1} \ldots A_1) \cdot p(A_{n-1} \mid A_{n-2} \ldots A_1) \cdots p(A_3 \mid A_2 A_1) \cdot \\ & p(A_2 \mid A_1) \cdot p(A_1) \end{aligned} \tag{2.3}$$

Totale Wahrscheinlichkeit und die Formel von Bayes

Im folgenden wird die Wahrscheinlichkeit eines Ereignisses berechnet, welches aus meheren Einzelexperimenten ausgewählt wird.

Definition 2.3. Es sei $A = \{a_1, \ldots, a_n\}$ die Menge der Elementarereignisse. Die Ereignisse $a_1, \ldots, a_n$ bilden eine *vollständige Ereignisdisjunktion*, wenn sie paaweise unverträglich sind, d. h. $\forall i, j \; i \neq j \quad a_i \cdot a_j = \varnothing$ und wenn bei jeder Versuchdurchführung eines der Ereignisse $a_1, \ldots, a_n$ eintritt.

Satz 2.4 (Totale Wahrscheinlichkeit). Es sei $a_1, \ldots, a_n$ eine vollständige Ereignisdisjunktion mit den Wahrscheinlichkeiten $p(a_i) > 0$ mit $\sum_{i=1}^{n} p(a_i) = 1$. Dann gilt für die Wahrscheinlichkeit eines beliebigen Ereignisses b

$$p(b) = \sum_{i=1}^{n} p(b \mid a_i) \cdot p(a_i) \tag{2.4}$$

Satz 2.5 (Formel von Bayes). Es sei $a_1, \ldots, a_n$ eine vollständige Ereignisdisjunktion mit $p(a_i) > 0$ für alle i. Für ein beliebiges Ereignis b mit $p(b) > 0$ gilt

$$p(a_j \mid b) = \frac{p(b \mid a_j) \cdot p(a_j)}{p(b)} = \frac{p(b \mid a_j) \cdot p(a_j)}{\sum_{i=1}^{n} p(b \mid a_i) \cdot p(a_i)} \; \textit{für } j = 1, \ldots, n. \tag{2.5}$$

Wahrscheinlichkeitsverteilung, Zufallvariable und Erwartungswert

Der Begriff *Wahrscheinlichkeitsverteilung* beschäftigt sich mit der Frage, wie sich beim Zufallexperiment die Wahrscheinlichkeiten auf verschiedene Ereignisse verteilt. Falls die Wahrscheinlichkeitsverteilung bekannt ist, dann kann festgestellt werden, welche Wahrscheinlichkeit diese Ereignisse bei dem betreffenden Zufallexperiment jeweils besitzen.
Wir interessieren uns für Zufallexperimente, bei denen jedem Ereignis pro Realisierung eine Zahl zugeordnet wird. Solche Zuordnung wird in folgender Definition beschrieben.

Definition 2.4. Eine *Zufallvariable* ist eine reellewertige Funktion X, welche jedem Ereignis bei pro Realisierung eines Zufallexperiments eine Zahl zuordnet. Die Wahrscheinlichkeit des Ereignisses „X nimmt den Wert a an“ wird mit $p(X = a)$ bezeichnet.

Diese Wahrscheinlichkeit unterliegt natürlich den Axiomen im Wahrschenilichkeitsraum $(A, \mathcal{P}(A), p)$. Je nach Anwendungsgebiet unterscheidet man zwischen *diskreten* oder *stetigen* Zufallvariablen. Wir interessieren uns für den Fall der diskreten Zufallvariablen, d. h. sie könnnen nur abzählbar endlich viele Werte annehmen.
Ist X eine Zufallvariable, die zu irgendeinem Zufallexperiment gehört, so existiert nach Definition zu jeder vorgegebenen Zahl a die Wahrscheinlichkeit $p(X = a)$.

Definition 2.5. Es seien $x_1, \ldots, x_n$ die Werte, für die X eine positive Wahrscheinlichkeit besitzt und $p_1, \ldots, p_n$ die zugehörigen Wahrscheinlichkeiten mit $p(X = x_i) = p_i$. Dann heißt die durch

$$f(x) = \begin{cases} p_i & \text{falls } x = x_i \\ 0 & \text{sonst} \end{cases} \tag{2.6}$$

definierte Funktion *Wahrscheinlichkeitsfunktion* der betreffenden Zufallvariable X.

Wegen $\sum_{x_j} f(x_j) = 1$, so kann man z.B. die Wahrscheinlichkeit $p(a < X \leq b)$ des Ereignisses $(a < X \leq b)$ einfach berechnen, falls die Wahrscheinlichkeitsfunktion f einer diskreten Zufallvariable bekannt ist. Es gilt also

$$p(a < X \leq b) = \sum_{a < X \leq b} f(x_j) = \sum_{a < x_j \leq b} p_j$$

Nun sei x nicht fest. Das Ereignis $X \leq x$ hat definitionsgemäß die Wahrscheinlichkeit $p(X \leq x)$. Da die Wahrscheinlichkeit von x abhängt, ist sie eine Funktion von x. Es folgt die

Definition 2.6. X sei eine Zufallvariable, die zu einem Zufallexperiment gehört. Dann heißt

$$F(x) = p(X \leq x)$$

die *Verteilungsfunktion* der betreffenden Zufallvariable X.

Wegen $p(X \leq x) = \sum_{x_j \leq x} f(x_j)$ so gilt für die Verteilungsfunktion einer diskreten Verteilung der folgende Satz

Satz 2.6. $f(x)$ sei die Wahrscheinlichkeitsfunktion einer diskreten Zufallvariable X. Dann gilt für die Verteilungsfunktion F

$$F(x) = \sum_{x_j \leq x} f(x_j) \tag{2.7}$$

Wir definieren nun die *(mathematische) Erwartung* einer Zufallvariable. Dazu seien $x_1, \ldots, x_n$ die Werte, für die X Wahrscheinlichkeit $p_1, \ldots, p_n$ mit $p(X = x_i) = p_i$ besitzt, d. h. eine Wahrscheinlichkeitsverteilung sei auf X gegeben. Die Erwartung der Zufallvariable X wird in folgender Definition skizziert

Definition 2.7. Die Erwartung einer diskreten Zufallvariable X ist definiert durch

$$\begin{aligned} E(X) &= \sum_j x_j \cdot p(X = x_j) \\ &= \sum_j x_j \cdot f(x_j) \end{aligned} \tag{2.8}$$

wobei $f(x)$ die Wahrscheinlichkeitsfunktion von X ist.

Für eine reellwertige Funktion $g(X)$, welche für alle möglichen Werte von X definiert ist, dann heißt

$$E\big(g(X)\big) = \sum_j g(x_j) \cdot f(x_j) \tag{2.9}$$

Erwartung von $g(X)$. Es ist zu beachten, daß $g(X)$ selbst eine Zufallvariable ist.

Bei Durchführung von Zufallexperimenten können mehrere Zufallgrößen als Zufallvariablen, die unabhängig voneinander eintreten. Sind X_1 und X_2 Zufallvariablen mit $p(X_1 = a) = p(a)$ und $p(X_2 = b) = p(b)$, und ist $p(X_1 = a \text{ und } X_2 = b)$ die Wahrscheinlichkeit, daß X_1 den Wert a und X_2 den Wert b annimmt. Die *(stochastische) Unabhängigkeit* von X_1 und X_2 wird charakterisiert durch

Definition 2.8. Die Zufallvariablen X_1 und X_2 und damit die zugehörigen Zufallexperimente heißen unabhängig, wenn gilt

$$\forall a, b \in \mathbb{R} \quad p(X_1 = a \text{ und } X_2 = b) = p(X_1 = a) \cdot p(X_2 = b) = p(a)p(b) \tag{2.10}$$

Gemäß dieser Definition gilt für die bedingte Wahrscheinlichkeit mit $p(b) > 0$

$$\begin{aligned} p(X_1 = a \mid X_2 = b) &= \frac{p(X_1 = a \text{ und } X_2 = b)}{p(X_2 = b)} \\ &= \frac{p(X_1 = a) \cdot p(X_2 = b)}{p(X_2 = b)} \\ &= p(X_1 = a) \end{aligned} \tag{2.11}$$

2.2.3 Grundbegriffe der Informationstheorie

Wir nehmen an, daß X eine Zufallvariable eines Zufallexperiments und $p(X)$ die Wahrscheinlichkeitsverteilung auf X. Es erhebt sich die Frage: Welche Information hat X bezüglich seiner Wahrscheinlichkeitsverteilung $p(X)$?
Um eine Antwort auf die Frage zu finden, betrachten wir ein Beispiel, bei dem *das Werfen einer Münze* durch X festgelegt. Die Wahrscheinlichkeitsverteilung auf X, d. h. die zugehörige Wahrscheinlichkeit für das Auftreten von *Kopf* bzw. *Zahl* beim Wurf der Münze beträgt $\frac{1}{2}$. Der Informationsgewinn an X beträgt 1 oder anders ausgedrückt: Die Ungewißheit vor dem Wurf einer Münze oder die Entropie $H(X)$ ist 1. Denn *Kopf* und *Zahl* lassen sich durch ein Bit kodieren, und zwar *Kopf* $= 1$ und *Zahl* $= 0$ oder umgekehr. Im folgenden betrachten wir den Begriff *Entropie* noch allgemeiner.

Information und Entropie

Als *Informationsgehalt* $I(U)$ für das mit der Wahrscheinlichkeit $p(U)$ eintretende Ereignis U definiert man $I(U) := \Leftrightarrow \log p(U)$ bzw. $I(a_i) := \Leftrightarrow \log p_i$ für das Elementarereignis a_i. Wählen wir den Logarihmus zur Basis 2, so wird der Informationsgehalt I in *Bits* gemessen. Mit der Formel $\log_b x = \log x \cdot \log b$ kann der Informationsgehalt

I für beliebige Basis b berechnet werden.
Für das *sichere Ereignis*, d. h. sein Eintrittwahrscheinlichkeit ist 1, ist der Informationsgehalt ist 0. Werden mehrere voneinander unabhängigen Ereignisse, die bei einem Zufallexperiment mit den Wahrscheinlichkeiten $p_1, \ldots, p_n$ eintreten, zu einem *gemeinsamen Ereignis* zusammengefaßt, so ist sein Wahrscheinlichkeit $p_1 \cdot p_2 \cdots p_n$. Sein Informationsgehalt beträgt dann $I = \Leftrightarrow \log(p_1 \cdot p_2 \cdots p_n) = \Leftrightarrow \sum_{i=1}^{n} \log p_i$.
Der Informationsgehalt ist um so größer, je kleiner ist die Eintrittwahrscheinlichkeit des betreffenden Ereignisses.
Ein wichtiges Hilfsmittel für die Sicherheitseinrichtung von Kryptosystemen ist die im Jahr 1948 von *Shannon* eingeführte *Theorie* , in der die *Entropie* ein Zentralbegriff ist.

Definition 2.9. Bezüglich aller Elementarereignisse a_i eines Wahscheinlichkeitsraums $(A, \mathcal{P}(A), p)$ wird der mittle Informationsgehalt als Entropie bezechnet. Es gilt die Gleichung für $n \geq 2$:

$$H(A) = H(p_1, \ldots, p_n) = \sum_{i=1}^{n} p(a_i) \cdot I(a_i) = \Leftrightarrow \sum_{i=1}^{n} p_i \cdot \log p_i \tag{2.12}$$

$H(A)$ ist nach Definition (xx) die Erwartung von Informationsgehalt $I(a_i)$. Die Entropie H gestaltet, die Ungewissheit eines Systems, z. B. Kryptosystems, zu bewerten. In der Kryptographie bedeutet, daß die Entropie zum Beispiel die zu ermittelnden Klartexbits angibt, mit denen die Bedeutung einer verschlüsselten Nachricht enthüllt werden kann. Mann kann auch sagen, daß die Entropie eines Kryptosystems ist ein Maß für die Größe des Schlüsselraums $\mathcal{K}$. Sie ist in der Praxis näherungsweise gegeben durch

$$H(\mathcal{K}) = \log |\mathcal{K}|,$$

wobei $|\mathcal{K}|$ die Mächtigkeit des Schlüsselraums ist. Wenn der Schlüssel in einem Kryptosystem zum Beispiel 64 Bit verwendet wird, so ist $|\mathcal{K}| = 2^{64}$. Die Entropie dieses Kryptosystems ist dann $H(\mathcal{K}) = \log 2^{64} = 64$ Bit. Daher ist Ein Kryptosystem umso schwieriger zu knacken, je größer seine Entropie ist.

Im folgenden werden einige Eigenschaften von Entropie wiedergegeben.

1. $H(A) \geq 0$
   $H(A) = 0 \Leftrightarrow \exists i \in \{1, \ldots, n\} : p(a_i) = 1$ und $\forall\, j \neq i : p(a_j) = 0$

2. Es seien:

   - $p_i \geq 0$ und $q_i > 0$,
   - $\sum_{i=0}^{n} p_i = \sum_{i=0}^{n} q_i = 1$.

   Dann gilt die Ungleichung

$$H(A) = H(p_1, \ldots, p_n) = \Leftrightarrow \sum_{i=1}^{n} p_i \log p_i \leq \Leftrightarrow \sum_{i=1}^{n} p_i \log q_i \tag{2.13}$$

3. Für ein festes n wird die Entropie maximal, wenn $p_i = \frac{1}{n}$ ist. Es gilt die Ungleichung:

$$\begin{aligned} H(p_1, \ldots, p_n) &\leq H\Big(\frac{1}{n}, \ldots, \frac{1}{n}\Big) \\ &= \log n \end{aligned} \tag{2.14}$$

   also insgesamt

$$0 \leq H(A) \leq \log n \tag{2.15}$$

4. Es gilt:

$$\begin{aligned} H(p_1, \ldots, p_n) = {} & H(p_1, \ldots, p_{i-1}, p_i + p_{i+1}, p_{i+2}, \ldots, p_n) \\ & + (p_i + p_{i+1}) H\Big(\frac{p_i}{p_i + p_{i+1}}, \frac{p_{i+1}}{p_i + p_{i+1}}\Big) \end{aligned} \tag{2.16}$$

5. und

$$H\Big(\underbrace{\frac{1}{nm}, \ldots, \frac{1}{nm}}_{nm-mal}\Big) = H\Big(\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}_{n-mal}\Big) + H\Big(\underbrace{\frac{1}{m}, \ldots, \frac{1}{m}}_{m-mal}\Big) \tag{2.17}$$

6. Die Entropie eines Systems wird nicht geändert, wenn man ein unmögliches Ereignis hinzunimmt. Das heißt

$$H(p_1, \ldots, p_n, 0) = H(p_1, \ldots, p_n)$$

Äquivokation (Rückschlußentropie)

Es seien X und Y zwei endliche Mengen mit

$$X = \{x_1, \ldots, x_m\} \qquad Y = \{y_1, \ldots, y_n\}.$$

Weiter bezeichnen wir:
$p(y_j|x_i)$, $i = 1, \ldots, m \quad j = 1, \ldots n$, ist die Wahrscheinlichkeit für das Eintreten von y_j, falls das Element x_i aus der Menge X gewählt wurde. In der Kryptographie kann interpretiert werden: $p(y_j|x_i)$ die Wahrscheinlichkeit dafür, daß y_j der Geheimtext ist, falls x_i der Klartext ist.
$p(x_i, y_j)$ ist die Wahrscheinlichkeit für das gemeinsame Eintreten von x_i und y_j.

$p(x_i)$ ist die *a priori Wahrscheinlichkeit* für das Auftreten von x_i und $p(y_j)$ die Wahrscheinlichkeit für y_j.
Nach dem *Satz über die totale Wahrscheinlichkeit* erhalten wir für $p(y_j)$:

$$p(y_j) = \sum_{i=1}^{m} p(y_j \mid x_i) \cdot p(x_i). \tag{2.18}$$

Durch Indexvertauschung erhalten wir auch

$$p(x_i) = \sum_{j=1}^{n} p(x_i \mid y_j) \cdot p(y_j). \tag{2.19}$$

Wegen

$$p(x_i|y_j) = \frac{p(x_i, y_j)}{p(y_j)}$$
$$p(y_j|x_i) = \frac{p(y_j, x_i)}{p(x_i)}$$

folgt

$$p(x_i, y_j) = p(y_j|x_i)p(x_i) = p(x_i|y_j)p(y_j) \tag{2.20}$$

und schließlich gilt nach der Formelvon *Bayes*

$$p(x_i \mid y_j) = \frac{p(y_j \mid x_i) \cdot p(x_i)}{p(y_j)} = \frac{p(y_j \mid x_i) \cdot p(x_i)}{\sum_{i=1}^{m} p(y_j \mid x_i) \cdot p(x_i)} \tag{2.21}$$

und

$$p(y_j \mid x_i) = \frac{p(x_i \mid y_j) \cdot p(y_j)}{p(x_i)} = \frac{p(x_i \mid y_j) \cdot p(y_j)}{\sum_{j=1}^{n} p(x_i \mid y_j) \cdot p(y_j)} \tag{2.22}$$

Die bedingte Wahrscheinlichkeit $p(x_i \mid y_j)$ nennt man *a posteriori Wahrscheinlichkeit*. Sie ist zum Beispiel die Wahrscheinlichkeit dafür, daß x_i der Klartext in der Situation, daß der Geheimtext y_j beobachtet wurde.

Definition 2.10. In diesem Zusammenhang wird die Entropie

$$H(X) = \Leftrightarrow \sum_{i=1}^{m} p(x_i) \cdot \log p(x_i)$$

als *a priori Entropie* von X und

$$H(X \mid Y = y_j) = \Leftrightarrow \sum_{i=1}^{m} p(x_i \mid y_j) \log p(x_i \mid y_j) \tag{2.23}$$

als *a posteriori Entropie* von X, falls y_j vorliegt. Sie kann als Maß für die Ungewißheit im Mittel betrachtet werden, wenn ein Element y_j bezüglich der Wahrscheinlichkeitverteilung $\{p(x_i \mid y_j)\}_{y_j \in Y}$ gewählt wurde, wenn also bekannt ist, daß das aus der Menge Y gewählte Element das Element y_j ist.
Die bedingte Entropie

$$H(X \mid Y) = \Leftrightarrow \sum_{j=1}^{n} p(y_j) H(X \mid Y = y_j) \tag{2.24}$$

heißt *Äquivokation* oder *Rückschlußentropie.*

Mann sieht: Die Rückschlußentropie ist die mathematische Erwartung der a posteriori Entropie. Die Rückschlußentropie kann auch geschrieben werden als

$$H(X \mid Y) = \Leftrightarrow \sum_{j=1}^{n} \sum_{i=1}^{m} p(y_j) p(x_i \mid y_j) \log p(x_i \mid y_j). \tag{2.25}$$

Für weitere Definitionen bezeichnet man

$$H(XY) = \Leftrightarrow \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i, y_j) \log p(x_i, y_j) \tag{2.26}$$

die Entropie der gemeinsamen Wahrscheinlichkeitverteilung $\{p(x_i, y_j)\}_{x_i \in X,\, y_j \in Y}$ von x_i und y_j. Für sie gilt:

$$H(XY) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y) \tag{2.27}$$

Beweis. Wir beweisen die Gleichung: $H(XY) = H(X) + H(Y \mid X)$

$$\begin{aligned}
H(X) + H(Y|X) &= \Leftrightarrow \sum_{i=1}^{m} p(x_i) \log p(x_i) \Leftrightarrow \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i) p(y_j|x_i) \log p(y_j|x_i) \\
&= \Leftrightarrow \Big\{ \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(y_j|x_i)} + \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i, y_j) \log p(y_j|x_i) \Big\} \\
&= \Leftrightarrow \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i, y_j) \log \Big\{ \frac{p(x_i, y_j) p(y_j|x_i)}{p(y_j|x_i)} \Big\} \\
&= \Leftrightarrow \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i, y_j) \log p(x_i, y_j) \\
&= H(XY)
\end{aligned}$$

Der Beweis anderer Gleichung verläuft ananlog.

Als Maß für die Abnahme der Ungewißheit eines Systems ist die *Transinformation.*

Definition 2.11. Die Transinformation $I(X;Y)$ ist definiert durch

$$I(X;Y) = H(X) \Leftrightarrow H(X \mid Y) \tag{2.28}$$

und hat folgende Eigenschaften:

(1) $$I(X;Y) = \Leftrightarrow\sum_{j=1}^{n}\sum_{i=1}^{m} p(x_i,y_j)\log\frac{p(x_i,y_j)}{p(x_i)p(y_j)},$$

denn es gilt nach Definition von Transinformation

$$\begin{aligned} I(X;Y) &= H(X) \Leftrightarrow H(X \mid Y) \\ &= \Leftrightarrow\sum_{i=1}^{m} p(x_i)\log p(x_i) + \sum_{j=1}^{n}\sum_{i=1}^{m} p(y_j)p(x_i \mid y_j)\log p(x_i \mid y_j) \end{aligned}$$

Mit dem Satz über totale Wahrscheinlichkeit, angewandt auf x_i, gilt

$$\begin{aligned} p(x_i) &= \sum_{j=1}^{n} p(y_j)p(x_i \mid y_j) \\ &= \sum_{j=1}^{n} p(y_j)\frac{p(x_i,y_j)}{p(y_j)} \\ &= \sum_{j=1}^{n} p(x_i,y_j) \end{aligned}$$

folgt

$$\begin{aligned} I(X;Y) &= \Leftrightarrow\sum_{i=1}^{m}\big(\sum_{j=1}^{n} p(x_i,y_j)\big)\log p(x_i) + \sum_{j=1}^{n}\sum_{i=1}^{m} p(y_j)p(x_i \mid y_j)\log p(x_i \mid y_j) \\ &= \Leftrightarrow\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i,y_j)\log p(x_i) + \sum_{j=1}^{n}\sum_{i=1}^{m} p(y_j)\frac{p(x_i,y_j)}{p(y_j)}\log p(x_i \mid y_j) \\ &= \Leftrightarrow\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i,y_j)\log p(x_i) + \sum_{j=1}^{n}\sum_{i=1}^{m} p(x_i,y_j)\log p(x_i \mid y_j) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i,y_j)\big\{\log p(x_i \mid y_j) \Leftrightarrow \log p(x_i)\big\} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i,y_j)\log\big\{\frac{p(x_i,y_j)}{p(x_i)p(y_j)}\big\} \end{aligned}$$

(2) $$I(X;Y) = I(Y;X)$$

(3) $$H(XY) = H(X) + H(Y) \Leftrightarrow I(X;Y)$$

(4) $$I(X;Y) \geq 0 \text{ und } I(X;Y) = 0 \Leftrightarrow \forall\, i,j: \quad p(x_i,y_j) = p(x_i)p(y_j)$$

Als Letztes geben wir noch die Definition von *bedingter Transinformation*

Definition 2.12. Es sei $Z = \{z_1, \ldots, z_l\}$. Die bedingte Transinformation sei definiert durch

$$I(X;Y \mid Z) = H(X \mid Z) \Leftrightarrow H(X \mid YZ), \tag{2.29}$$

wobei $H(X \mid YZ) = \sum_{j=1}^{n} p(y_j) H\big(X \mid (Y = y_j)Z\big)$ mit

$H\big(X \mid (Y = y_j)Z\big) = \Leftrightarrow \sum_{i=1}^{m} \sum_{k=1}^{l} p\big((x_i, z_k)|y_j\big) \log p\big(x_i|(y_j, z_k)\big)$ für ein $j \in \{1, \ldots, n\}$ ist.

Zusammengefaßt ist die bedingte Transinformation gegeben durch

$$I(X;Y|Z) = \Leftrightarrow \sum_{k=1}^{l} \sum_{i=1}^{m} p(z_k) p(x_i|z_k) \log p(x_i|z_k) \tag{2.30}$$
$$+ \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{k=1}^{l} p(y_j) p\big((x_i, z_k)|y_j\big) \log p\big(x_i|(y_j, z_k)\big)$$

Die bedingte Transinformation hat folgende Eigenschaften:

$$I(X;Y|Z) \geq 0 \tag{2.31}$$
$$I(X;Y|Z) = I(Y;X|Z) \tag{2.32}$$
$$I(X;YZ) = I(X;Z) + I(X;Y|Z) \tag{2.33}$$

2.2.4 Grundbegriffe aus der Informationstheorie

2.3 Komplexität

In diesem Abschnitt werden nur die Bit-Speicherung der Zahlen sowie Bit-Operation zusammengefaßt. Die Bit-Komplexität in $\mathbb{Z}$. Es sei Anzahl der Bits für die binäre Darstellunh von n $\lfloor \log n \rfloor + 1 \approx \log n$. Dann gilt für die Bit-Komlexität:

Operation		Bit-Komplexität
Addition	$a + b$	$O(\log n)$
Subtraktion	$a \Leftrightarrow b$	$O(\log n)$
Multipklikation	$a \cdot b$	$O((\log n)^2)$
Division	$a = qb + r$	$O((\log n)^2)$

Analog ist die Bit-Komplexität in $\mathbb{Z}_n$ gegebne durch folgende Tabelle

Operation		Bit-Komplexität
Addition	$(a + b) \mod n$	$O(\log n)$
Subtraktion	$(a \Leftrightarrow b) \mod n$	$O(\log n)$
Multipklikation	$(a \cdot b) \mod n$	$O((\log n)^2)$
Multipklikation	$a^k \mod n,\ k < n \mod n$	$O((\log n)^3)$
Division	$a^{-1} \mod n$	$O((\log n)^2)$

2.4 Kryptographische Grundlagen

2.4.1 Kryptosysteme

Definition 2.13. Es seien

- $\mathcal{M}$ ein Klartextraum,
- $\mathcal{C}$ ein Chiffretextraum,
- $\mathcal{K}$ ein Schlüsselraum,
- $\mathcal{E} = \{e_K \mid e_K : \mathcal{M} \times \mathcal{K} \leftrightarrow\!\!\!\rightarrow \mathcal{G}, K \in \mathcal{K}$ und e_K injektiv$\}$ Menge der Verschlüsselungsfunktionen und
- $\mathcal{D} = \{d_K \mid d_K : \quad \mathcal{C} \times \mathcal{K} \leftrightarrow\!\!\!\rightarrow \mathcal{M}, K \in \mathcal{K}\}$ Menge der Entschlüsselungsfunktionen.

Dabei sind $\mathcal{M}, \mathcal{C}$ und $\mathcal{K}$ höchstens abzählbar. Dann heißt der 5-Tupel $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein *kryptographisches System (kurz: Krytosystem)*, wenn für jedes $K \in \mathcal{K}$ eine e_K und zugehörige d_K gibt, so daß $d_K(e_K(x)) = x$ für jedes $x \in \mathcal{M}$ gilt.

Kryptosysteme müssen folgenden Eigenschaften erfüllen:

1. Ver- und Entschlüsselungsfunktion müssen für jeden Schlüssel $K \in \mathcal{K}$ effizient berechnet werden.
2. Das Krytosystem muß benutzerfreundlich sein, d.h. es ist einfach einen K sowie die Funktion e_K und d_K zu finden.
3. Die Sicherheit des System soll auf der Geheimhaltung des Schlüssels, jedoch nicht auf der Geheimhaltung der verwendeten Algorithmen beruhen. Mit anderen Worten aus der Kenntnissen von e_K bzw. d_K kann der Klartext nicht abgeleitet werden.

Für die Geheimhaltung und authentizität gibt es spezielle Anforderungen: Geheimhaltung verlangt, daß es dem Kryptoanalytiker nicht möglich ist, Klatext aus Geheimtext abzuleiten. Authetizität verlangt, daß der Kryptoanalytiker ohne Aufdeckung keinen anderen Geheimtext C' für einen Geheimtext C einsetzen kann. Im folgenden werden diese Anforderungen auführlich beschrieben:

Geheimhaltungsanforderung

Es soll einem Kryptoanylytiker berechnungmäßig praktisch unmöglich sein,

1. systematisch d_K aus abgefangenen Geheimtext C zu bestimmen, auch wenn der Klartext M mit $e_K(M) = C$ bekannt ist und

2. den klartext M aus C zu gewinnen.

Für die Geheimhaltung muß nur e_K geschützt werden, wenn d_K nicht aus e_K abgeleitet werden kann. e_K kann somit veröffentlicht werden. **Authetizitätsanforderung**

Es soll dem Kryptoanylytiker berechnungmäßig praktisch unmöglich sein,

1. systematisch e_K aus C zu bestimmen, wenn der Klartext M mit $e_K(M) = C$ bekannt ist oder

2. einen anderen Geheimtext C' zu finden, so daß $d_K(C')$ ein gültiger Klartext aus $\mathcal{M}$ ist.

Durch die Anforderung 1. wird sichergestellt, daß der Kryptoanalytiker keinen falschen Klartext M' durch $e_K(M') = C$ einschieben kann. Für die Authetizität muß nur e_K geschützt werden, wenn d_k nicht e_k verrät. d_k kann somit veröffentlicht werden.

Zusammengefaßt, kann man feststellen, daß der Schlüsselt K in einen Chiffrier- bzw. Dechiffrierschlüssel aufgeteilt werden kann. Wenn beiden Schlüssel gleich sind, erhält man ein *sysmetrisches Kryptosystem*. Dieses Kryptosystem ist gut geeignet für private Ionformation, die über ein Netzwerk übertragen werden.
Sind die beiden Schlüssel von einander verschieden, so daß ein Schlüssel aus den anderen nicht abgeleitet werden kann, dann spricht mann von einem asymmetrischen oder Public-Key-Kryptosystem. Die wichtigste Eigenschaft eines Public-Key-Kryptosystem ist: Man kann eine der beiden Transformationen veröffentlichen, ohne die verbleibende zu gefährden. Diese Eigenschaft wird in der Praxis beispielsweise zur Sicherung von Datenbanken benutzt. Demnach besitzt der Benutzer bei der „read-only authority“ d_k, bei der „write-only authority“ besitzt er e_K.

Im Rahmen dieser Diplomarbeit gehen wir nicht in die sysmetrische Kryptosysteme ein, sondern aus Bequemlichkeit nur in die Public-Key-Kryptosysteme. Zwei typische Public-Key-Kryptosysteme sind RSA und ElGamal. Diese Kryptosysteme werden im Abschnitt **Signatur** behandeln, da sie sowohl für die Verschlüsselung als auch für Signierung eingesetzt werden können.

2.4.2 Sicherheit von Kryptosystemen

Eine der wichtigen Eigenschaften von Kryptosystemen ist ihre Sicheheit unter allgemeinen Annahmen zu beweisen. In dem Design von Kryptosystemen bedeutet Die Güte eines Kryptosystems ist ihr praktischer Einsatz und ihre Sicherheit. Es ist manchmal ein Abwgen (*trade-off*) zwischen den beiden, denn ein sehr sicheres Kryptosystem kann in der Praxis kaum eingesetzt werden und ist ineffizient , z.B. *one time pad*. Im allgemeinen besteht das Sicherheitmedell von Kryptosystemen aus zwei Basisansätzen. Das sind *computationale* und *unconditionale* Sicherheit [30].

2.4.2.1 Uneingeschänkte Sicheheit (*unconditional security*)

Ein Kryptosystem heißt *uneingeschänkt sicher*, wenn es bei unbegrenzten Rechenressourcen nicht zu brechen ist. uneingeschränkt sichere Kryptosysteme werden in der Literatur auch Kryptosysteme mit perfekter Geheimhaltung (*perfect secrecy*) oder perfekt-sichere Kryptosysteme in der Literatur [30] bezeichnet. Die perfekte Geheimhaltung besagt, daß der Klartext statistisch vom Chiffretext abhängig ist [26] ab hängig ist. Dies bedeutet, daß in einem perfekt-sicheren Kryptosystem ein Angreifer nicht anders tut, als den Klartext aus der Kenntnis des Chiffretextes zu raten und zwar unabhängig davon, wieviel Rechnhenkapazität und rechenzeit er zur Verfügung hat. Ein One-Time-Pad ist zum Beispiel uneingeschänkt sicher.
In einem *symmetrischen Kryptosystem* mit perfekter Geheimhaltung muß der verwendete Schlüssel genauso lange wie der Klartext sein. Dagegen kann ein *Public-Key*-Kryptosystem nicht unconditional-sicher sein, da der Klartext des vorgegebene Chiffretextes C „im Prinzip“ rekonstruiert werden kann, indem man alle mögliche Klartexte verschlüsseln, bis ein Chiffretext C^* mit $C^* = C$ gefunden worden ist. Die Untersuchung auf perfekte Sicherheit kann nicht vom Standpunkt der *computationalen* Sicherheit vorgenommen werden, da wir für sie nur beschränkte Rechenkapazität in Frage kommt.

Mit dieser Bemerkung werden *unconditional-sicher (perfekte) Secret Sharing Schemes* daher auf die Grundlagen von Wahrscheinlichkeits- und Informationstheorie, insbesondere Entropie, untersucht.

2.4.2.2 Berechenbare und beweisbare Sicheheit (*computational security*)

Praktisch-berechenbare Sicherheit (computational Security) basiert auf die verfügbare Rechenkapazität, durchdie ein Kryptosystem mit einem besten Algorithmus mit mindestens N Operationen, wobei N ein sehr große Zahl ist, gebrochen werden kann. In diesem Sinn heißt ein Kryptosystem *berechnungssicher (computational secure)*, wenn die bestmöglichen Angriffe die Fähigkeit eines Kryptoanalytiker übertrifft. Leider kennt man weder den besten Algorithmus noch die besten Angriffe. Die Sicherheit eines Krytyptosystems kann daher nicht von diesem Standpunkt bewiesen werden. In der Praxis muß man sie durch bekannten Algorithmen bzw. Angriffsverfahren approxomieren. So heißt in der Praxis ein Kryptosystem berechnungssicher, wenn es mit der heutzutage verfügbaren Rechenkapazität nicht in vernünftiger Berechnungszeit gebrochen werden kann. Solche Systeme sind zum Beispiel RSA, Elgamal, DES, IDEA, RC5, es sind einige zu nennen.
Anderer Ansatz für die praktisch-berechnebare Sicherheit ist die *beweisbare Sicherheit* (*provably secure*) [30]. Man versucht die Sicherheit eines Kryptosystems mit dem Lösen schwieriger mathematischen Probleme zu vergleichen. Das sind zum Beispiel das Faktorisieren großer Zahlen und Berechnung des diskreten Logarithmus in bestimmten Gruppen. So kann man approximativ sagen: Ein gegebene Kryptosystem ist sicher, wenn die im System verwendete Zahl nicht faktorisiert oder die Berechnug des diskreten Logarithmus in der gegebenen Gruppe nicht durchgeführt werden

kann. Obwohl es bis heute nicht bewiesen ist, daß das Brechen eines Kryptosystems genau so schwierig ist wie das Faktorisieren einer großen Zahl oder die Berechnung des diskreten Logarithmus, gehen meistens Kryptosystemen von diesem Ansatz aus. Man sagt: Die Sicherheit solcher Systeme basieren auf *unbewiesene kryptographische Annahmen.*

In der Arbeit werden auch *berechnungssichere Secret Sharing Schemes* betrachtet, da sie im Vergleich mit traditionellen Schemata flexibel und effizient sind.

2.4.3 Kryptoanalyse

Die *Kryptologie (cryptology)* ist die Wissenschaft des Verbergen von Wörtern. Sie setzt sich aus *Kryptographie (cryptography).* Die Kryptographie befaßt sich mit der Entwicklung von Kryptosystemen, durch deren Verschlüsselungstranformation e_K Informationen in andere Form gebracht werden, so daß sie für Unbefugten nicht verständlich ist. Wie diese Kryptosysteme arbeiten, wurde in der Definition 2.13 auf der Seite **??** beschrieben.

Im Rahmen der Kryptoanalyse werden Kryptosysteme in Bezug auf ihren Schutz vor kryptoanalytische Angriffen untersucht und beurteilt. Dies ist sogar ein Bestandteil von der Entwicklung sicherer Kryptosysteme, die zunächst gegen kryptoanalitische Angriffsmethoden widerstandsfähig sein sollten. Die Sicheheit eines Kryptosystems basiert in der Praxis einerseits darauf, daß man zunächst gegnerische Kryptosysteme zu brechen versucht, um seine eignen als sicherer zu beweisen. Es besteht aus der Kryptoanalyse andererseits ein Problem, daß sie auf das Vorhandensein eines kryptographischen Gegenstand beruht, während kryptographische Angriffe aus menschlichen Phantasien ausgehen. Aus diesem Grund braucht ein sicheres Kryptosystem wirksame kryptoanalytische Fähigkeit des Gegenspielers. Ein Kryptograph wird immer gut daran tun, die Fähigkeit seines Gegenspielers, also die Kryptoanalytiker, nicht zu unterschätzen. Dies ist eine der Empfehlungen von Shannon beim Entwurf von Kryptosystemen. Diese Tatsache begründet sich durch ein Beispiel, daß die Firma RSA Data Security, inc. Wettbewerbe offiziell ins Leben gerufen. Dabei wurde beispielsweise die RSA-129, das ist 129-stellige Zahl (426 Binärstellen), 1994 von Lenstra zusammen mit 600 freiwilligen Helfern auf 1.600 Workstations faktorisiert (*factoring by e-mail*). Heutzutage RSA-Kryptosystem mit dem Modul von 155-stelliger Zahl (512 bit)ist üblich. Es wird sogar empfohlen, daß für RSA grössere Module, z.B. 1024 bit einzusetzen sind.

Im Bezug auf Kryptanalysis versucht ein Kryptanalist bzw. Angreifer (ein der beiden wird in Abhängigkeit vom Standpunkt verwendet) zum Beispiel eine Verschlüsselung dadurch zu brechen, daß er aus einem Chiffretext den zugehörigen Klartext zu ermitteln oder daß er den für die Verschlüsselung verwendeten Schlüssel bzw. aus einem Klar- und Chiffretext-Paar den verwendeten Schlüssel zu finden.

Theoretisch kann ein Kryptanalist statistische und mathematische Abhängigkeit aus den Daten ermitteln, die er beispielsweise von Kommunikationsteilnehmern gewonnen hat. Solche mathematische Verfahren und Methoden der Kryptanalysis werden

in [13] beschrieben.
Die *praktische Kryptanalysis* versucht, die Schlüssel oder auch die geheimzuhaltenden Nachrichten ohne Anwendung mathematischer Methoden zu ermitteln (Siehe CB94). Die Überlistung der Benutzer gehört ebenso dazu wie die Manipulation von Hardware und Software. Gegen die praktische Kryptanalysis müssen administative Maßnahmen, wie physische Zugangskontrolle und Schulung der Benutzer, ergriffen werden.

Im folgenden werden einige typische Angriffsarten diskutiert. Das sind Angriffe auf kryptographische Verfahren und Protokolle

2.4.3.1 Angriffe auf die kryptographischen Verfahren

Kryptographische Verfahren werden üblicherweise so ausgelegt, daß die im folgenden beschriebenen Angriffe erfolglos bleiben.

Viele der Verfahren basieren auf der praktischen Unlösbarkeit bestimmter mathematischer Probleme. So basiert RSA auf der Schwierigkeit der Faktorisierung großer Zahlen, der Diffie Hellman Schlüsselaustausch auf der Schwierigkeit der Berechnung des diskreten Logarithmus.

Solange diese Probleme nicht effizient gelöst werden können, muß der Angreifer für die Lösung eines solchen Problems extrem hohen Zeitaufwand verwenden. Damit läßt sich durch lineares Vergrößern der verwendeten Zahlen der Aufwand für einen Angreifer um ein Vielfaches erhöhen. Die einzigen erfolgreichen Angriffe auf ein starkes Verfahren bleiben die praktische Kryptoanalysis und das Ausprobieren aller möglichen Schlüssel.

Exhaustive Search, Brute Force
Im Zusammenhang mit der *praktische Sicherheit* (*securrity*) eines Kryptosystems untersucht man ein in der Praxis oft verwendeten Angriffsart. Das ist der *Brute-Force-Angriff*, der auch als *vollständige Schlüsselsuche* (*exhaustive key search*) bezeichnet wird. Ein Kryptosystem heißt *praktisch-sicher (secure)*, wenn der bestbekannteste Angriff nicht einfacher ist als die vollständige Schlüsselsuche. Bei einem Brute-Force-Angriff wird der gesamte Schlüsselraum durchgesucht. Dies bedeutet, daß Angriefer jeden möglichen Schlüssel aus dem Schlüsselraum zur Entschlüsselung eines Chiffretextes heran zu kommen versuchen. Der Brute-Force-Angriff funktioniert mit Kryptosystemen, beiden es zu jedem Chiffretext höchstens einen Klatext gibt, und nur möglich, wenn midestens ein Klax- und Chiffretext-Paar den Angreifern bekannt ist. Will man sich vor Brutte-Force-Angriffen schützen, so muß der im System verwendete Schlüsselraum hinrechend groß gemacht und der verwendete geheime Schlüssel aus diesem Raum *zufällig* gewählt wird. Leider ist die *zufällige Auswahl* der Schlüsselwerte in praktischer Implemtierung keine einfache Aufgabe.
Dabei ist allerdings zur Kenntnis zunehmen, daß das Durchsuchen des Schlüsselraums praktisch von mehreren Rechnern parallel durchgeführt werden kann. Die Zeit zur Ermittlung eines Schlüssels ist mit den verfügbaren Finanzmitteln des Angreifers und mit der Leistung der Rechner verbunden.
Das abschließende Beispiel zeigt, daß der von it RSA Data Security ausgeschriebene

Wettbewerb, gennannt *DES-Challenge (siehe dazu Lab97)* ruft dazu auf, DES neben anderen Kryptosystem durch einen Brute-Force-Angriff zu brechen, ist nicht grundlos.
Beim DES ist der Schlüsselraum mit $2^{56} = 72\ 057\ 594\ 037\ 927\ 936$ Schlüsseln zu klein, um ernsthafte Angriffe groer Firmen abzuwehren. Ein Rechner, der zum Beispiel eine Million Schlüssel pro Sekunde ausprobieren kann, benötigt durchschnittlich $\frac{1}{2} \cdot \frac{2^{56}}{10^6} \approx 3,6 \cdot 10^{10}$ Sekunden. Das sind ca. 1150 Jahre. Um diesen Schlüsselraum durch zu suchen, benötigen diese 1150 Rechner nur noch ein Jahr. Der schnellste Rechner hat im Schnitt 3.9 Millionen Schlüssel pro Sekunde ausprobiert.

Birthday Attack.
Dieser Typ von Angriffen finden in *digitalen Signatur-Schemata* mit verwendung von *Einweg-Hashfunktionen* statt. Der Angriff basiert auf die Tatsache aus der Statistik - auch *Birthday Paradoxon* genannt: In einer Personengruppe (23 Personen) ist die Wahrscheinlichkeit, daß jemand am gleichen Tag Geburtstag hat wie eine gegebene Person vergleichweise klein, während die Wahrscheinlichkeit, daß zwei beliebige Personen am gleichen Tag Geburtstag haben, ist relativ hoch ($p \geq \frac{1}{2}$). Daher ist es einfacher für einen Angreifer, zwei beliebige Nachrichten zu finden, die den gleichen *Hashwert* haben, als eine Nachricht, die den gleichen Hashwert hat, wie die vorgegebene Nachricht.
Der erste Angriff heißt *Angriff auf 2nd-Preimage* und der zweite *Angriff auf kollisionsresistente Einweg-Hashfunktionen*. Ist der Hashwert 160 bit lang, so benötigt Brute-Force-Angriff auf 2nd-Preimage 2^{160} Hashoperationen, während bei dem zweiten, also für die Suche nach einer Kollision, *nur* 2^{80} Hash-Operationen [30].

Ciphertext-only-Angriffe
Bei diesem Angriff ist der kryptanalytische Angreifer im Besitz von Chiffretexten. Sein Ziel ist, chiffretexte durch irgendein Verfahren zu rekonstruieren oder den verwendeten Schlüssel zu ermitteln, damit die nächste Chiffretexte mit dem gefundenen Schlüssel dechiffriert werden können. Angenommen, die Verschlüsselungsfunktion sei E_K und es gebe verschiedene verschlüsselte Textsegmenten $C_1 = E_K(P_1), C_2 = E_K(P_2), \ldots, C_i = E_K(P_i)$. Im Rahmen eines Ciphertext-only-Angriffe verfügt der kryptanalytische Angreifer über die Ciphertexte C_i $(i > 1)$. Er versucht dann entweder mit einem bestimmten Algorithmus die Klartexte P_i zu rekonstruieren oder den schlüssel K daraus zu extrahieren, damit er mit dem Schlüssel K den Klatext P_{i+1} aus $C_{i+1} = E_K(P_{i+1})$ erhalten kann. Dieser Angriffsart kommt häufig in der Praxis vor, da es immer möglich ist, verschlüsselte Daten über öffentliches Kommunikationskanal abzuhören.

Known-Plaintext-Angriffe
Bei diesem Angriff hat der kryptanalytische Angreifer Chiffretexten und Zusätzlich die zugehörigen Klartexte. Auch hier können die Chiffretexte das Ergebnis des Abhörens von verschlüsselten Daten sein, die über öffentliches Kommunikationskanal gesendet worden sind. Er versucht dann aus $(C_i, E_K(P_i))$ den nächsten P_{i+1} aus $(C_{i+1} = E_K(P_{i+1}))$ zu rekonstruieren oder den Schlüssel K zu extrahieren . Falls $i = 1$ ist, so führt der Angreifer eine *exhaustive key search* (Bruto-Force-Angriff) mit $2^{|\mathcal{K}|}$ Operationen durch, wobei $|\mathcal{K}|$ die Anzahl aller möglichen Schlüsselwerte von K.

Chosen-Plaintext-Angriffe
Beim Chosen-Plaintext-Angriffe kennt der kryptoanalytischer Angreifer nicht nur die Klartexte P_i und Chiffretexte $(C_i, E_K(P_i))$, sondern er kann sie selbst generieren oder verschlüsseln. Mit Hilfe der Verschlüsselungsfunktion kann er aus P_i entsprechende Chiffretexte $(C_i, E_K(P_i))$ bilden. Bei diesem Angriff hat der Angreifer das Ziel, ein Verfahren oder einen Schlüssel K zu finden, um den Klatext P_{i+1} aus dem Chiffretext $(C_{i+1} = E_K(P_{i+1}))$ abzuleiten.

Chosen-Ciphertext-Angriffe
Im Rahmen eines Chosen-Plaintext-Angriffs hat der kryptoanalytischer Angreifer Chiffretexte $(C_i, E_K(P_i))$ und den Zugang zu der Verschlüsselungsfunktion. Er kann beliebig gewählten Chiffretexte entschlüsselt lassen. Mit Hilfe der Verschlüsselungsfunktion erhält er die Klartexte P_i aus $(C_i, E_K(P_i))$. Ziel dieses Angriffs ist die Ableitung des Schlüssel K, damit der nächste Klartext P_{i+1} aus dem Chiffretext $(C_{i+1} = E_K(P_{i+1}))$ gewonnen werden kann. Chosen-Ciphertext-Angriffe spielen überwiegend bei asymmetrischen Verschlüsselungen eine Rolle. denoch hier Chiffretext wählen, den er entschlüsseln lassen kann.

adaptive Chosen-Plaintext- und Chosen-Ciphertext-Angriffe
Besteht es eine Auswahl zwischen dem zu chiffrierenden und dem zu dechiffrierenden Text in Abhängigkeit von einem vorhandenen Chiffretext, der gebrochen werden soll, so spricht man von einem adaptive Chosen-Plaintext- und Chosen-Ciphertext-Angriff.

2.4.3.2 Angriff auf kryptographische Protokolle

Kryptographische Protokolle werden immer zwischen zwei oder mehr Parteien durchgeführt. Bei Angriffen auf kryptographische Protokolle kann der Angreifer die verschiedensten Zwecke verfolgen:

- Ausforschen der Schlüsselinformation,
- Personenidentifizierung einer Kommunikationspartei oder
- Unterbrechnung des Protokollablaufs.

Kryptographische Protokolle sollten so entworfen werden, daß sie sicher gegen die beschriebenen Angriffsarten sind. Wird dennoch ein solcher Angriff gestartet, sollte das Protokoll dies bemerken und abgebrochen werden. Weiterhin sollte ein Angreifer aus den im Laufe des Protokolls produzierten Daten keine Erkenntnisse erlangen können.

Passives Abhören der Übertragung
Dies ist der einfachste Angriff auf ein kryptographisches Protokoll. Der Angreifer schreibt alle übertragenen Informationen auf und wertet sie für spätere Verwendung aus.

Man-in-the-Middle-Angriffe
Hier schaltet sich der Angreifer zwischen den Parteien, welche ein kryptographisches Protokoll gerade ausführen. Er kann beispielsweise die Public-Key einer Partei durch

seinen ersetzen, um an der Ausführung des Protokolls zu beteiligen, ohne daß die Parteien merken. Weiterhin kann er die Übertragungen verändern, blockieren. Dies ist das Problem von Protokollen, welche exponentielle Funktionen basieren. Der Man-in-the-Middle-Angriff kann zum Beispiel auf Schlüsseltausch-Protokoll ausgeübt werden. Falls die Authentifikation von den beiden öffentlichen Informationen $y_A = \alpha^{x_A} \mod p$ und $y_B = \alpha^{x_B} \mod p$ im Schlüsseltausch-Protokoll von Diffie und Hellman nicht gesichert sind ($x_A, x_B \in \mathbb{Z}_p$ sind jeweils selbst von A und B ausgedachte Zahlen, p ist prim, α primitives Element), dann kann ein Angreifer E auf dem Übertragungskanal zwischen A und B sowohl x_A, als auch x_B durch seine öffentliche Information $y_E = \alpha^{x_E} \mod p$ ersetzen. Dann teilt er den Schlüssel $K_1 = \alpha^{x_A x_E} \mod p$ der Person A und $K_2 = \alpha^{x_B x_E} \mod p$ der Person B mit. Der Angreifer E kann nun alle zwischen A und B übertragenbnen Nachrichten mit den beiden Schlüsseln K_1 und K_2 entsclüsseln, lesen, verändern und wieder verschlüsseln, ohne daß dies von der jeweiligen Person A und B aufgedeckt werden könnte.
Dieser Angriff kann aber verhindert werden, wenn man die Authentikation von $y_A = \alpha^{x_a} \mod p$ und $y_B = \alpha^{x_B} \mod p$ durch Zertifikate absichern. Die Authetikation jedes öffentlichen Schlüssels wird durch *Certification Authority* digital unterschrieben.

Impersotnaion, Maskade (Verkörperung einer Partei)
Der Angreifer versucht hier die Identität einer Partei anzunehmen und so die andere Partei dazu zu bringen, geheime Informationen zu übermitteln.

Replay und Cut-and-Paste (Wiederverwendung von Teilnachrichten)
Beim Wiedereinspielen benutzt der Angreifer Teile zuvor abgehörter Nachrichten. Diese Teile - er kann ihren Inhalt kennen oder auch nicht - benutzt er nun bei Abwicklung eines Protokolls.
Beim **Cut-and-Paste-Angriffen** kennt der Angreifer den genauen Inhalt nicht, beispielsweise wenn der Inhalt mit einem dem Angreifer nicht bekannten Schlüssel chiffriert ist. Er verwendet Nachrichtenteile gewissermaßen blind.
Beim **Replay-Angriffen** verwendet der Angreifer für ihn lesbare Informationen, die er aus anderen Protokollabläufen oder aus abgehörter Kommunikation gewonnen hat, fur späteren Zeitpunkte.

2.4.4 kryptographische Primitiven

2.4.4.1 krytographische Einweg-Hashfunktion

Hashfunktionen spielen in der Informatik und modernen Kryptographie eine wichtige Rolle. Hashfunktionen werden in *Digitale Signatur-Schemata* und Anwendungen, in denen man Datenintegrität und Nachtenauthentikation fordert, eingesetzt. *Hashfunktionen* werden ausführlich in der Literatur diskutiert. In vielen Fällen sind die zu signierenden Daten groß sind, ist es ratsam, nur ihre Hashwerte zu signieren. Durch Einsatz von Hashfunktion kann man eine Nachricht beliebiger Länge in *Message Digest* der festgelegten transformieren. Message Digests, aus denen die Signatur mittels eines Signatur-Schema erzeugt wird, müßen fälschunssicher sein. Die Fälschungen können durch Angriffe auf die verwendete Hashfunktion hevorgerufen werden.

Da die durch ein Secret Sharing Schema erzeugten *Shares* groß sein können und über ein unsicheres Kanal authentisch an den *Schlüsselspeicher* bzw. die *Wächter*, die für Rekonstruktion des Geheimnises verantwortlich sind, gesendet werden müßen. müßen sie neben zusätzlicher Verschlüsselung signiert werden. Wir diskutieren hier nur wichtige Eigenschaften der Hashfunktion, ihre Sicherheit und Auswahl der für die Signatur verwenden Hashfunktionen. Wir beginnen mit der Definition einer Hashfunktion.

Definition 2.14. Eine Funktion $H : \mathcal{M} \Leftrightarrow\!\!\!\rightarrow \mathcal{M}_H$ heißt Hashfunktion, wenn die beiden Eigenschaften haben:

- Jede Nachricht beliebiger Länge aus $\mathcal{M}$ wird in binäre Bitfolge fester Länge (*Hashwert*) abgebildet.
- Für eine beliebige Eingabe $m \in \mathcal{M}$ ist es, effizient seinen Hashwert $H(m)$ zu berechnen.

Dabei ist $\mathcal{M}$ der Nachrichtenraum und $\mathcal{M}_H$ die Menge der Hashwerte In der Kryptographie braucht man Hashfunktionen mit weiteren Eigenschaften, die für kryptographische Anwendungen erforderlich sind.

Eigenschaft von Hashfunktion

1. Zu einem gegebenen Hashwert $h \in \mathcal{M}_H$ ist es *praktisch nicht möglich*, ein $m \in \mathcal{M}$ zu finden, so daß $h = H(m)$.
2. Zu einer gegebenen Nachricht $m \in \mathcal{M}$ mit dem zugehörigen Hashwert $H(m) \in \mathcal{M}_H$ ist es *praktisch nicht möglich*, eine zweite Nachricht $m' \in \mathcal{M}$ zu finden, so daß $H(m') = H(m)$.
3. Es ist *praktisch nicht möglich*, zwei beliebige Nachrichten $m', m \in \mathcal{M}$ mit $m' \neq m$ zu finden, so daß $H(m') = H(m)$

Die erste Eigenschaft heißt *einweg*, die zweite *kollisionsfrei* und die dritte *kollisionsresistent*. In der Literatur werden diese Eigenschaften auch *preimage resistent*, *2nd-preimage resistent* und *kollisionsresistent*. Im Zusammenhang mit Birthday-Angriffe ist die dritte Eigenschaft eine Verschärfung der zweiten.

Hashfunktion in der 1.1 Definition mit der Eigenschaft 1 und 2 wird als *Einweg-Hashfunktion* oder *schwache Einweg-Hashfunktion* bezeichnet.

Kollisionsresistente Hashfunktion ist Hashfunktion in der 1.1 Definition mit den Eigenschaften 2 und 3. Diese Hashfunktionnen heißen auch *starke Einweg-Hashfunktion*.

Sicherheit von Hashfunktionen und typische Anfriffe

Ist ein Hashwert h der Länge n vorgegeben, so beträgt die Wahrscheinlichkeit für

die Suche nach einem m, so daß $H(m) = h$ gilt, 2^{-n}. Ein Bruto-Force-Angriff auf Hashfunktion erfordert somit 2^n Hash-Operationen. Um gegen solche Angriffe abzuwehren, soll der verwendete Hashwert mindestens 128 Bits lang sein.

Der zweite Angriff auf schwache Einweg-Hashfunktion ist Bruto-Force gegen die 2nd-preimage Resistance. Der Angriff geschieht in der folgenden Situation: Person P_1 will die Nachricht m signieren. sie bildet zuerst den Hashwert $h = H(m)$ und signiert auf $H(m)$. Die Signatur von m ist $s = \mathsf{Sig}_d(h)$. P_1 schickt $(m, \mathsf{Sig}_d(h))$ zu P_2. Die Person P_2 bildet nicht $r = H(m)$, um die Gleichung $\mathsf{Ver}_e(r, s) = true$ zu prüfen, sondern sucht eine m' mit $H(m') = H(m)$. P_2 behauptet dann, daß P_1 die Nachricht m' signiert hätte. Um diesen Angriff durchführen zu können, muß P_2 insgesamt 2^n Hash-Operationen ausführen.

Der dritte Angriff gegen kollisionsresistente Hashfunktion ist der Birthday-Angriff. Dieser Angriff gilt als gefährlich, da er schneller als Brutto-Force-Angriff durchgeführt werden kann. Beim Birthday-Angriff geht es darum, daß ein Angreifer zwei beliebige Nachrichten m' und m mit $H(m') = H(m)$ zu finden versucht. Damit der Angriff erfolgreich durchgeführt werden kann, muß der Angreifer näherungsweise $2^{\frac{n}{2}}$ Operationen ausführen. Haswerte der Länge 64 Bits sind daher zu klein. Aus diesem Grund arbeitet die sichere Hashfunktion, basierend auf den sicheren Hash-Algorithmus (*Secure Hash Algorithm*), mit 160 Bits. Diese Hashfunktion ist ein Bestandteil des digitalen Signatur-Standards (DSS).

Hashfunktionen mit unterschiedlichen Eigenschaften, deren Sicherheit und typische Angriffe fassen wir in folgender Tabelle.

Typ	Sicherheit	Stärke	Angriffsziel
Einweg-Hashfunktionen	. einweg	2^n	$h = H(m)$ geg., suche nach m
	. kollisionsfrei	2^n	Suche nach 2-tem m' mit $H(m') = H(m)$
Kollisionsresistente Hashfunktionen	kollisionsresistent	$2^{\frac{n}{2}}$	Suche nach Kollision

Bisher haben wir nur „*Unkey-Hashfunktionen*“ behandelt. Wird die Hashwerte zusätzlich gesichert, so spricht man von *Message Authentication Codes* MAC. Das ist ein spezialgebiet von Hashfunktionen. *MAC* wird nicht nur in Signatur-Schemata verwendet.

2.4.4.2 Signatur-Verfahren

Mit digitalen Signaturen kann Authentizität der Nachricht gesichert werden. Es gibt zwei Arte von Signatur: Signatur mit Nachrichtrekonstruieren und Signatur mit anhängender Nachricht.

Bei Signatur mit Nachrichtrekonstruktion (*digital signature giving message recovery*) wird die Nachricht im Klartext nicht mitgesendet, sondern nur die Signatur. Bei der Verifikation der Signatur wird die Nachricht rekonstruiert.
Bei dem Einsatz von Signatur mit Anhang (*digital signature with appendix*), wie der Name sagt, muß die Nachricht im Klartext zusammen mit der Signatur gesendet werden.
Wir achten darauf, daß die Hashfunktion bei den hier zumschluß ausgewählten Signatur-Schemata verwendet wird (wegen Speicherplatz(, da die Shares im Slüsselspeicher liegen müssen).

Wir beschreiben zuerst allgemein, wie ein Signatur Schema arbeitet. Danach werden wir digitale Signatur Schemata, die auf Public-Key-Kryptosysteme (RSA, Elgamal) basieren sowie Signatur Schemata, die für unsere Anwendung im 5. Kapitel beschreiben : (unbestreitbare, modifizierte Rabin Signatur-Schemata). Ob wir ferner Signatur-Schema mit Nachrichtrecovery-Eigenschaft benutzen, hängt davon ab, ob die redundante Funktion sicher und effizient ist. Wir analysieren auch: die Geschwindigkeit und Parameterwahl.

Es seien $\mathcal{M}$ ein endlicher Nachrichtenraum, $\mathcal{S}$ eine endliche Menge der Signaturen, $\mathcal{K}$ ein Schlüsselraum und $\mathcal{M}_{mathcalS}$ bzw. $\mathcal{M}_{\mathcal{V}}$ endliche Menge der verwendeten Signatur- bzw. Verrifikationsalgorithmen.

Definition 2.15. Ein (digitales) signatur schema ist ein 5-Tupel $(\mathcal{M}, \mathcal{S}, \mathcal{K}, \mathcal{M}_{\mathcal{S}}, \mathcal{M}_{\mathcal{V}})$ mit der Eigenschaft:

- $\forall\ K \in \mathcal{K}\ \exists$ einen SignierAlgorithmus $\mathsf{Sig}_K \in \mathcal{M}_{\mathcal{S}}$ und einen zugehörigen Verifikationsalgorithmus $\mathsf{Ver}_K \in \mathcal{M}_{\mathcal{V}}$.
- Für jede Funktion $\mathsf{Sig}_K : \mathcal{M} \Longleftrightarrow \mathcal{S}$ und $\mathsf{Ver}_K : \mathcal{M} \times \mathcal{S} \Longleftrightarrow \{ja, nein\}$ gilt : $\forall\ m \in \mathcal{M}$ und $s \in \mathcal{S}$

$$\mathsf{Ver}(m, s) = \begin{cases} ja & \text{falls } s = \mathsf{Sig}(m) \\ nein & \text{falls } s \neq \mathsf{Sig}(m). \end{cases}$$

Dabei ist m die zu signierende Nachricht, s die Signatur und Sig_K bzw. Ver_K der vom Schlüssel K abhängigen Signier- bzw. Verifikationsalgorithmus. In der Definition sind K und $\mathsf{Sig}_K(\cdot)$ geheim. $\mathsf{Ver}_K(\cdot)$ ist alledings öffentlich. Die Auswertung der beiden Funktionen sind haben polynomielle Zeitkomplexität.

In jedem Signatur Schema ist wüschenswert, daß eine Signatur s auf m fälschungssicher sein sollte. Falls Bob beispielsweise der Unterzeichner ist, könnte nur er die die Signatur s auf m mit Hilfe von berechen, so daß $\mathsf{Ver}(m, s) = ja$ gilt. Und es wäre recherisch undurchführbar, Bobs Signatur auf m zu fälschen. Doch kann ein Signatur-Schema nicht uneingeschränkt (*unconditional secure*) sicher sein, denn der Angreifer Oscar kann anhand der öffentlichen Funktion $\mathsf{Ver}(\cdot)$ alle möglichen Signaturen s für m testen, bis die passende Signatur gefunden ist. Wäre die Suchzeit für Oscar vertretbar, so könnte Bobs Signatur in diesem Fall gefälscht werden.

Ziel des Entwurfs von Signatur-Schemata ist daher, daß sie „*computational*“ sicher sind. Unter Zuhilfenahme bekannter kryptographischen (unbewiesenen) Annahmen, z.B. Faktorisierung einer großen Zahl n, Berechnug des diskreten Logarithmus, sowie kryptographischen Primitiven, z.B. kollisionsresistente Hashfunktionen, können computationale Signatur-Schemata konstruiert werden.

Durch Einsatz von Hashfunktionen in Signatur-Schemata werden nicht nur die zu signierenden Nachrichten beliebiger Länge zu Nachrichten fester Länge, mindestens 128 Bits, reduziert, sondern auch die Fälschung der Signatur. Mit der Verwendung von Hashfunktion wird 1.2 Definition wie folgt erweitert:

Definition 2.16. Sei zusätzlich $H : \mathcal{M} \Longleftrightarrow \mathcal{M}_H$ eine (kolissionsresistente) Hashfunktion mit $\mathcal{M}_H = \{H(m) \mid m \in \mathcal{M} \text{ und } |H(m)| \geq 128\, bits\}$ der Wertebereich von H. Ein *computationales* Signatur-Schema ist ein 6-Tupel $(\mathcal{M}, \mathcal{S}, \mathcal{K}, \mathcal{M}_\mathcal{S}, \mathcal{M}_\mathcal{V}, \mathcal{M}_H)$ mit der Eigenschaft:

- $\forall\, m \in \mathcal{M}$ kann $H(m)$ effizient berechnet werden
- $\forall\, K \in \mathcal{K}\ \exists$ einen Signieralgorithmus $\mathsf{Sig}_K \in \mathcal{M}_\mathcal{S}$ und einen zugehörigen Verifikationsalgorithmus $\mathsf{Ver}_K \in \mathcal{M}_\mathcal{V}$.
- Für jede Funktion $\mathsf{Sig}_K : \mathcal{M}_H \Longleftrightarrow \mathcal{S}$ und $\mathsf{Ver}_K : \mathcal{M} \times \mathcal{S} \Longleftrightarrow \{ja, nein\}$ gilt : $\forall\, m \in \mathcal{M}$ und $s \in \mathcal{S}$

$$\mathsf{Ver}(m, s) = \begin{cases} ja & \text{falls } s = (\mathsf{Sig} \circ H)(m) \\ nein & \text{falls } s \neq (\mathsf{Sig} \circ H)(m). \end{cases}$$

In der 1.3 Definition ist die verwendete Hashfunktion H öffentlich. Statt m wird der Hashwert der Länge beispielsweise 128 Bits mit der Funktion $\mathsf{Sig}(\cdot)$ signiert und $\mathsf{Ver}(\cdot)$ kann die Echtheit der Signatur geprüft werden.
Will Bob zum Beispiel die Nachricht m authentisch an Alice senden. Zuerst bildet er $h = H(m)$. Dann signiert er h. Die Nachricht m und die Signatur $s = \mathsf{Sig}(h)$ sendet er an Alice. Alice berechnet $r = H(m)$ und vergleicht das Erbebnis mit s. Die Signatur ist korrekt, falls $s = r$ ist.
Um die Bobs Signatur zu fälschen, muß Alice eine Nachricht m' finden, so daß $H(m') = H(m)$ gilt, d.h. Alice muß 2^{128} Hashoperation ausführen (*2nd preimage attack*).
Mittels Birthday-Angriff kann Bob aber auch Alice täuschen, indem er 2^{64} Nachrichten untersucht, um eine Kollision heraus zu finden, d.h. er nimmt zwei beliebige Nachrichten $m, m' \in \mathcal{M}$ mit $H(m') = H(m)$. Nachdem er gefunden hat, läßt Bob Alice auf den $H(m)$ signieren. Später kann Bob Alice überzeugen, daß sie $H(m')$ unterschriben hätte.

Ähnlich wie 1.3 Definition lassen sich mit Hilfe von RSA und ElGamal asymmetrische Signatur-Schemata konstruieren. Sie sind computational sicher, da ihre Sicherheit auf die Faktorisierungsproblem großer Zahl und Undurhführbarkeit der Berechnug von diskreten Logarithmen basieren.

RSA Signatur-Schema

Es seien $n = pq$, p und q prim, $\mathcal{M} = \mathcal{S} = \mathbb{Z}_n$. Der Schlüsselraum sei $mathcalK = \{(n, p, q, e, d) \mid n = pq \text{ und } ed \equiv 1 \mod \phi(n)\}$. Die Werte n und e sind öffentlich. p, q d sind geheim.
Für $K = (n, p, q, e, d)$ definiere für $m, s \in \mathbb{Z}_n$

1. die Signaturfunktion $\mathsf{Sig}_K = x^d \mod n$ und
2. die Signaturfunktion $\mathsf{Ver}_K(m, s) = ja \Leftrightarrow x \equiv s^e \mod n$

Mit dem RSA Signatur-Schema kann die Nachricht m wie folgt signiert werden. Dazu sei (e_{Bob}, d_{Bob}) das Public-Key-Paar von Bob und (e_{Alice}, d_{Alice}). Angenommen, Bob möchte die Nachricht m signiert, verschlüsselt an Alice senden. Bob signiert die Nachricht m, indem er $s = \mathsf{Sig}_{e_{Bob}}(m)$. Dann verschüsselt er m und s beide mit $C = ENC_{e_{Alice}}(m, s)$. Der Chiffretext C sendet er an Alice. Nachdem Alice C erhalten hat, entschlüsselt sie den Chiffretext ihn mit d_{Alice} und bekommt (m, s). Mit Hilfe der öffentlichen Verifikationsfunktion kann Alice überprüfen, ob $\mathsf{Ver}_{e_{Bob}}(m, s) = ja$ gilt.

Verschlüsselt Bob jedoch erst die Nachricht m und den Chriffretext mit seinem Privat-Schlüssel. Die Signatur sendet er an Alice. In diesem Fall kann Oscar mit dem „Man-in-the-Middle-Angriff“ arbeiten, um die Bobs Signatur fälschen, ohne daß Alice etwas davon weiß. Es ist daher empfohlen, daß man die Nachricht zuerst signiert dann verschlüsselt.

ElGamal Signatur-Schema

ElGamal Signatur-Schema ist ebenfalls asmmyetrisches Signatur-Schema, dessen Sicherheit auf die Undurchfürhrbarkeit der Berechnung von diskreten Logarithmen. Dieses Schema wird modifiziert und benutzt für digitales Signaturstandard (DSS). Das ElGamal Signatur-Schema läßt folgendes beschreiben:

Es sei p eine Primzahl deart, daß das diskrete Problem in $\mathbb{Z}_p$ undurchführbar ist, α das primitives Element in $\mathbb{Z}_p$ und $\mathcal{M} = \mathbb{Z}_p^*$, $\mathcal{S} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$.

1. Definiere $\mathcal{K} = \{(p, \alpha, \beta, a) \mid \beta \equiv \alpha^a \mod p\}$. a ist geheim und p, α und β sind öffentlich.
2. Für $K = (p, \alpha, \beta, a)$ und ein geheim zufällig gewähltes $k \in \mathbb{Z}_{p-1}^*$ definiere

   $$\mathsf{Sig}_K(m, k) = (\gamma, \delta)$$

   mit

   - $\gamma = \alpha^k \mod p$ und
   - $\delta = (m \Leftrightarrow a\gamma)k^{-1} \mod (p \Leftrightarrow 1)$

3. Für $m, \gamma \in \mathbb{Z}_p^*$ und $\delta \in \mathbb{Z}_{p-1}$ definiere

   $$\mathsf{Ver}_K(m, \gamma, \delta) = ja \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^m \mod p.$$

Zusammengefaßt, falls nicht anders verlangt wird, wird dieses Sisnatur-Verfahren für die Lösung im Kapitel 5 benutzt, um die Authetizität der Shares bei der Übermittlung vom Submitter an Schlüsselspeicher zu grantieren. Und zusammen mit probabilistischen Publib-Key-Kryptosystem wird zusätzlich die Authentikation gesichert. Diese beiden Ergebnisse sind die Modifikation von Secret Sharing Schemata, die im Kapitel 4 beschrieben sind und im Kapitel 5 verwendet werden.

Das digitale Signatur-Standard (DSS)
Ein Signatur-Schema ist ein Tripel von drei effizienten Algorithmen (Key-Gen, Sig, Ver).

- Key-gen ist ein Algorithmus, mit dem das Öffentlich/Privat-Schlüsselpaar (e, d) generiert wird. Mit dem Privaten Schlüssel d signiert der Sender $(signer)$ eine Nachricht m. Der Empfänger verifiziert die signierte Nachricht mit dem öffentlichen Schlüssel e des Senders, um die Echtheit der Nachricht zu bestätigen.
- Sig ist der Signatur-Algorithmus. Als Eingaben sind die Nachricht m und der private Schlüssel d. Wird der Algorithmus vollständig ausgeführt, so wird die Signatur $\mathsf{Sig}_d(m)$ aus gegeben.
- Ver ist der Verifikationsalgorithmus mit den Eingaben $e, m, \mathsf{Sig}_d(m)$ und der Ausgaben *yes* oder *no*.

In Anlenung an [10] und von [18] benutzen wir folgende Notation fr das Signatur-Schema, da aufgrund gewissen Anforderungen Sitzungsschlüssel für die Signatur erzeugt werden müssen.

Key-Generation. Der öffentliche und private Schlüssel werden mit dem Algorithmus Key-Gen plus zusätzliche Informationen generiert:

- p ist eine Primzahl der Länge l mit $512 \leq l \leq 1024$, l ist ein Mehrfach von 64.
- q ist weitere Primzahl der Länge 160 Bits mit der Eigenschaft $q|(p \Leftrightarrow 1)$.
- g ist ein Element der Ordnung q in $\mathbb{Z}_p^*$.
- Der private Signaturschlüssel d ist ein zufällige Zahl $1 \leq d < q$.
- Der öffentliche Verifikationsschlüssel ist $e = g^d \mod p$.

Die Zahlen p, q und g sind öffentlich. Da der Berechnung vom diskreten Logarithmus in vertretbarem Zeitaufwand nicht durchführbar ist, bleibt der private Schlüssel d in $e = g^d \mod p$ geheim.

Signatur. Kolisionsresistenten Hashfunktionen werden hier eingesetzt. Es sei $H(m)$ der Hashwert der Nachricht m. Der Sender signiert die Nachricht m mit folgenden Schritten:

- Der Sender wählt zufällig ein k mit $1 \leq k < q$ und berechnet $k^{-1} \mod q$.

- Weiter Berechnet er

$$\begin{aligned} r =& (g^{k^{-1}} \mod p) \mod q \\ s =& k(H(m) + dr) \mod q. \end{aligned}$$

Die Signatur (r, s) ist dann die Ausgabe des Signatur-Algorithmus Sig.

Verifikation. Der Empfänger kann die Signatur (r, s) anhand des öffentlichen Verifikationsschlüssels $e = g^d \mod p$ durch die Gleichung

$$r = (g^{H(m)s^{-1}} e_s^{rs-1} \mod p) \mod q$$

überprüfen, wobei $s^{-1} \mod q$ zu berechnen ist.

Wir haben bisher Signatur-Schemata mit Anhang (*Sisgnature Schemes with Appendix*) behandelt. Es gibt Signatur-Schemata mit der Fähigkeit, daß die eigentlichen Nachrichten, die signiert sind, nicht zum Empfänger mitgesendet werden. Diese Nachricht werden bei der Verifikation wiederhergestellt. Dazu sind sog. *Redundanzfunktion* erforderlich. Das Finden solcher Funktionen ist nicht einfach (siehe zum Beispiel Signature Schemes von giving Message recovery von Rabin).

Außerdem gibt es spezielle Signatur-Schemata für zahlreichende Anwendungen, die im Rahmen dieser Arbeit nicht behandelt werden kann. das Sind zum Beispiel unbestreitbare , Blind-Signatur-Schemata etc.

wenn die zu signierenden Nachrichten zu lange sind, wird passende Hashfunktion eingesetzt, um die Nachricht auf feste Länge ab. Es wird nur auf den Hashwert der Nachricht signiert.

2.4.4.3 Authentisierung im öffentlichen Netzwerk

Durch sichere Signature-schemata werden die Authentizität und Integrität Nachrichten gesichert. Im erahmen der *Authetisierung (genauer: Benutzerauthentisierung)* indentifizieren sich die beiden Kommunikationspartner gegenseitig, bevor sie miteinader über ein offenes Netzwerk miteinander kommunizieren wollen.
In einem Multiuser-Plattform, zum beispiel UNIX, wird die Authentisierung der Benutzer durch *Benutzerkennung (Benutzername)* und Kennwort *Password* ermöglich. Die eingegebenen Passwords werden mittels einer Einweg-Hashfunktion *gehashed* und im System gepeichert. Diesen Authentisierungsvorgang wird als *schwache Authentisierung (weak authentication)* bezeichnet. Da die Arbeit start auf kryptographischen Aspekt basieren möchte, sollen anderen Authentisierungstechniken eingesetzt werden. Das sind Identifikationsschemata, die als *Challenge-Response-Indentifikation Schemes* oder *strong authentication* bezeichnet. Solche Schemata werden im Kapitel 5 benutzt bzw. in erforderliche Protokolle eingebettet.

Die wichtigsten Identifikationsschemata sind von Fiat und Shamir [8, 9]. Mit sehr geringerem Kommunikationsaufwand ist das Schema von Schnorr [20], welches von T. Okamoto [19] modifiziert wurde, damit das Schema beweisbar sicher ist. Ein schnelles

Identifikationsschema durch Password von L. Lamport, allerdings nicht sicher, ist in [17].
Alle o.g. Schemata benötigen eine *vertrauenswürdige Zertifizierungstelle (certificat authority – CA)*, die für *Zertifikatenausgabe* zuständig ist. In einem Zertifikat ist u.a. die durch CA signierten öffentlichen Schlüssel eines Benutzer im Netz enthalten. Die Zertifikate können die Benutzer aus einer öffentlichen Datenbank hohlen.
Als Alternative für die o.g. Schemata ist das sicheres 3-*Weg-Authetisierungsprotokoll* nach *X.509 Standard* [34] verwendet werden.

Kapitel 3

Zugriffsstrukturen

3.1 Terminologie

Wir bezeichen das zu rekonstruierende Geheimnis als *Key* und das Teilgeheimnis als *Share*. Die Menge aller möglichen Werte, die der Key K annehmen kann, wird *Schlüsselraum* genannt. Weiter bezeichen $\mathcal{S}$ die Menge der Shares. Allgemein gilt: $|\mathcal{S}| \geq |\mathcal{K}|$.

Bei jedem Secret Sharing Schema geht man von einer endlichen Menge von Wächtern, bezeichnet als $\mathcal{W} = \{W_1, \ldots, W_n\}$, und einem *Dealer* D aus, $D \notin \mathcal{W}$. Jedes Secret Sharing Schema besteht aus zwei Prozessen: Die Verteilung der Shares und Rekonstruktion des Key. Die entscheidende Fragen sind: Wer darf den Key rekonstruieren und wie wird der Key rekonstruiert. In diesem Kapitel beanworten die erste Frage, indem wir die Strukturen der Wächter untersuchen. Die Antwort auf die zweite Frage geben wir im 4. Kapitel.

Welche Wächter aus $\mathcal{W}$ den Key $K \in K$ rekonstruieren dürfen, werden durch die *Zugriffsstruktur* festgelegt. Eine Zugriffsstruktur $\mathcal{A} \subseteq 2^{\{W_1, \ldots, W_n\}}$ auf $\mathcal{W}$ ist eine Familie von Teilmengen von Wächtern. Eine Zugriffsstruktur legt fest, welche Teilmengen von $\mathcal{W}$ den Key rekonstruieren können. Solche Teilmengen werden autorisierte Teilmengen genannt. Eine Festlegung der Zugriffsstrukturen auf $\mathcal{W}$ haben Wirkungen auf die Sicherheit, Effizient und Flexibilität des secret Sharing Schemas, das sie realisiert. Zugriffsstrukturen können unterschiedliche Eigenschaften haben.

Sei $B \in \mathcal{A}$ eine autorisierte Teilmenge, d.h. die Wächter können mit ihren Shares den Key K rekonstruieren. Betrachten wir eine Teilmenge C mit $B \subseteq C \subseteq \mathcal{W}$. Diese Teilmenge kann den Key K durch Ignorieren der Shares der Wächter in $C \setminus B$ rekonstruieren, d.h. C ist eine autorisierte Teilmenge und muß in $\mathcal{A}$ enthalten sein. Zugriffsstruktur mit dieser Eigenschaft führt zu

Definition 3.1 (Monotone Zugriffsstruktur). Sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Men-

ge von Wächtern. Eine Zugriffsstruktur $\mathcal{A}$ heißt *monoton*, falls gilt:

$$A \in \mathcal{A},\ A \subseteq B \subseteq \mathcal{W} \Longrightarrow B \in \mathcal{A}$$

Bei eine monotonen Zugriffsstrukturen ist also jede Teilmenge von Wächtern, die bereits eine autorisierte Teilmenge enthält, ebenfalls autorisiert. In der Arbeit werden nur monotone Zugriffsstrukturen benötigt und es wird vorausgesetzt, daß eine Zugriffsstruktur mindestens eine autorisierte Teilmenge enthält, d.h. $\mathcal{A} \neq \emptyset$. Eine Zugriffsstruktur $\mathcal{A}$ heißt genau dann *trivial*, wenn entweder $\mathcal{A} = 2^{\mathcal{W}}$ oder $\mathcal{A} = \{\mathcal{W}\}$

Enthält eine autorisierte Teilmenge außer sich keine weiteren autorisierten Teilmengen, so besitzt die Zugriffsstruktur eine weitere Eigenschaft.

Definition 3.2 (Basis). Sei $\mathcal{A}$ eine Zugriffsstruktur. Dann heißt die Teilmenge $B \in \mathcal{A}$ *eine minimal autorisierte* Teilmenge, wenn gilt:

$$A \subseteq B,\ A \neq B \Rightarrow A \notin \mathcal{A}.$$

Die Menge von minimal autorisierten Teilmengen heißt *Basis* von $\mathcal{A}$ und wird mit $\mathcal{A}^0$ bezeichnet.

Da die Zugriffsstruktur alle autorisierten Teilmengen aus $\mathcal{W}$ enthält, das sind die Obermengen einer Teilmenge in $\mathcal{A}^0$, kann die Zugiffsstruktur $\mathcal{A}$ als Funktion von der Basis $\mathcal{A}^0$ eindeutig bestimmt werden. Um es zu klären, wird der *Abschluß* einer Menge definiert.

Definition 3.3 (Abschluß). Sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ und $\mathcal{T} \in 2^{\mathcal{W}}$. Der *Abschluß* von $\mathcal{T}$, bezeichnet als $\mathsf{cl}(\mathcal{T})$, ist definiert durch die Menge

$$\mathsf{cl}(\mathcal{T}) = \{C \mid \exists\, B \in \mathcal{T} :\ B \subseteq C \subseteq \mathcal{W}\}$$

Ist $\mathcal{W} = \{W_1, \ldots, W_n\}$ und $\mathcal{A}^0$ eine Basis, dann ist $\mathcal{A} = \{C \subseteq \mathcal{W} \mid B \subseteq C$ und $B \in \mathcal{A}^0\}$ der *Abschluß* von $\mathcal{A}^0$. Es gilt daher $\mathcal{A} = \mathsf{cl}(\mathcal{A}^0)$. Die Zugriffsstruktur $\mathcal{A}$ kann als Funktion von der Basis $\mathcal{A}^0$ dargestellt werden.
Durch das folgende Beispiel wird der Zusammenhang zwischen $\mathcal{A}$ und $\mathcal{A}^0$ veranschaulicht.
Sei $\mathcal{W} = \{W_1, \ldots, W_4\}$ und $\mathcal{A}^0 = \big\{\{W_1, W_2, W_4\}, \{W_1, W_3, W_4\}, \{W_2, W_3\}\big\}$ eine Basis.
Dann ist

$$\mathcal{A} = \mathcal{A}^0 \bigcup \big\{\{W_1, W_2, W_3\}, \{W_2, W_3, W_4\}, \{W_1, W_2, W_3, W_4\}\big\}$$

eine Zugriffsstruktur auf $\mathcal{W}$ mit der Basis $\mathcal{A}^0$.
Umgekehr ist diese Zugriffsstruktur gegeben, so gilt offensichtlich, daß die Basis nur aus minimal autorisierten Teilmengen in der Zugriffsstruktur besteht.
Als Folgerung gilt: Ist die Zugriffsstruktur $\mathcal{A}$ monoton, dann gilt: $\mathcal{A} = \mathsf{cl}(\mathcal{A})$.

In [12] wird $\mathcal{A}_\mathcal{P} = \{B \subseteq \mathcal{W} \mid BautorisierteTeilmenge\}$ als *positive* und $\mathcal{A}_\mathcal{N} = \{B \subseteq \mathcal{W} \mid Bnicht\Leftrightarrow autorisierteTeilmenge\}$ als *negative Zugriffsstruktur* bezeichnet. Dabei gilt: $\mathcal{A}_\mathcal{P} \cap \mathcal{A}_\mathcal{N} = \emptyset$ und $\mathcal{A}_\mathcal{P} \cup \mathcal{A}_\mathcal{N} = 2^\mathcal{W}$. Diese Bezeichnung haben Lin und Harn eingeführt, da ihre Untersuchung auf Betrug-Probleme von negativen Zugriffsstrukturen ausgeht.

3.2 Threshold-Zugriffsstruktur

Definition 3.4 ((k,n)-Threshold-Zugriffsstruktur). Sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas und sei k eine nichtnegative ganze Zahl mit $k \leq n$. Eine Zugriffsstruktur $\mathcal{A} \subseteq 2^\mathcal{W}$ mit

$$\mathcal{A} = \{B \subseteq \mathcal{W} \mid |B| \geq k\}$$

heißt (k,n)-*Threshold-Zugriffsstruktur*. Der Parameter k heißt *Schwellenwert* (Threshold value).

In einer Threshold-Zugriffsstruktur gibt es mindestens $\binom{n}{k}$ autorisierte Teilmengen, die den Key rekonstruieren. Die Basis $\mathcal{A}^0$ enthält $\binom{n}{k}$ minimal autorisierten Teilmengen, die jeweils exakt k Wächtern enthalten. Daher ist die Basis der Theshold-Zugriffsstruktur ist $\mathcal{A}^0 = \{B \subseteq \mathcal{A} \, |B| = k\}$.
Wenn der Key K durch ein Secret Sharing Schema, das die Threshold-Zugriffsstruktur realisiert, rekonstruiert wird, so heißt es *(k,n)-Threshold-Schema*. Eigenschaften von (k,n)-Threshold-Schema diskutieren wir zusammen neben anderen Secret Sharing Schemata im 4-ten Kapitel. Bezüglich der vorliegenden Threshold-Zugriffsstruktur bemerken wir die Bedeutung der in der Zugriffsstruktur verwendeten Parameter n und k sowie ihre Wechselwirkungen, die wir für die nächsten Kapitel in der Arbeit verwenden. Secret Sharing Schemata, die die Threshol-Zugriffsstruktur für Rekonstruktion des Key realisieren, haben zwei Eigenschaften: Redundanter Rekonstruktion des Key K und Sicherheit gegen kryptographische Angriffe auf Geheimhaltung des Key K. Wir können uns allerdings für eine von beiden entschieden, und zwar in der Abhängigkeit davon, welche Anforderungen an das Verfahren gemacht werden sollen.

- Threshold-Zugriffsstruktur erlaubt, daß höchstens $(n \Leftrightarrow k)$ vernichtet bzw. beschädigt werden können. Wird $k \ll n$ gewählt, so ist die Redundanz für Rekonstruktion des Key hoch, d.h. der Key K kann der Key immer noch rekonstruiert werden, wenn fast so viel wie n Shares verloren gehen. Die Sicherheit des Schemas muß schwere Einbuße erleiden, da Angreifer leichter zu den k Shares gelangen, um den Key K zu rekonstruieren.

- Wird der Parameter k deart gewählt, daß $(n \Leftrightarrow k) \ll n$ gilt, dann ist eine illegale Rekonstruktion des Key schwerer. Denn Angreifer müssen fast n Shares im Besitz sein. Das Verfahren ist bei dieser Parameterwahl ist sicher und die

redundanter Recovery-Eigenschaft des Schemas kann erhalten bleiben, wenn geeignete Maßnahme gegen sukzessive Vernichtung bzw. Beschädigung bis zu $(n \Leftrightarrow k)$ getroffen wird.

Im 4-ten Kapitel wird gezeigt, daß noch weitere Maßnahmen getroffen werden mußsen, um die Sicherheit des Secret Sharing Schemas zu verbessern, da die Wächter in realistischen Anwendungen gegenseitig betrügen könnten.

Es gibt Situation, in der die Hierarchie im reellen Leben eine Rolle spielt. Dazu ist eine Zugriffsstruktur, in der jeder autorisierten Teilmenge ein Level zugeordnet ist. Der Key kann nur rekonstruiert werden, wenn entweder gewisse Anzahl der Wächter in einem Level oder gewisse Anzahl der Wächter in einer Level zusammen mit gewisser Anzahl der Wächter niedriges Levels ihre Shares frei geben. Formal werden solche Zugriffstrukturen folgende Definition beschrieben:

3.3 Multilevel-Zugriffsstrukturen

Definition 3.5 (Multilevel-Zugriffsstruktur). Sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas und sei $L = \{l_1, \ldots, l_r\}$ positiver ganzen Zahlen, bezeichnet als Levels mit $l_1 < l_2 < \cdots < l_r$, $l_{i \in \{1,2,\ldots,r\}} \neq 0$. Ferner sei $l : \mathcal{W} \Leftrightarrow\!\!\rightarrow L$ eine Abbildung, die jedem Wächter W ein Level $l(W) \in L$ zuordnet.

Eine Zugriffsstruktur $\mathcal{A} \subseteq 2^{\{W_1,\ldots,W_n\}}$ mit

$$\mathcal{A} = \left\{ B \subseteq W \middle| \exists \text{ ein } k : \left| \{W \in B : \ l(W) \leq k\} \right| \geq k \right\}$$

wird als $(l_1, \ldots, l_r)$-*Multilevel-Zugriffsstruktur* bezeichnet.

Die Zuordnung jedes Wächters zu einem Level entspricht seiner bestimmten Kompetenz. Die Menge der Wächter wird in r Gruppen $U_1, \ldots, U_r$ geteilt, so daß n_i die Anzahl der Wächter in der Gruppe U_i mit $\left| \bigcup_{i=1}^{r} U_i \right| = \sum_{i=1}^{r} n_i = n$ gilt. n_i Wächtern in Gruppe U_i wird ein Level $l_i \leq n_i$ $(i \in \{1, 2, \ldots, r\})$ zugeordnet. Aurhorisiert ist eine Teilmenge, in der mindestens l_s Wächter des Levels l_s oder hoher Levels $(l_1, \ldots, l_{s-1})$ „partizipieren“ (beteiligen). Wächter im Level l_s können gegenseitig ersetzen bzw. ein Wächter im Level l_s kann auch als Wächter im Level l_t mit $t > s$ agieren.
Um die Definition zu veranschaulichen betrachten wir eine $(2, 3)$-Multilevel-Zugriffsstruktur. Es seien $\mathcal{W}_7 = \{W_1, \ldots, W_7\}$ eine Menge von 7 Wächter, $|U_1| = 3$, $|U_2| = 4$. Den Wächtern aus $U_1 = \{W_1, W_2, W_3\}$ wird das Level $l_1 = 2$ und aus $U_2 = \{W_4, W_5, W_6, W_7\}$ das Level $l_2 = 3$ zugeordnet. Dann zulässige Konstallationen, also autorisierte Teilmengen von $\mathcal{W}_7$, sind zum Beispiel:

- entweder mindestens 2 Wächter aus U_1 (und kein Wächter aus U_2) oder

- 1 Wächter aus U_1 und mindestens 2 Wächter aus U_2 oder
- (kein Wächter aus U_1) und mindestens 3 Wächter aus U_2

können den Key bestimmen. Nicht-autorisierte Teilmengen sind zum Beispiel $\{W_1\}, \{W_4\}$, $\{W_4, W_5\}, \{W_1, W_4\}$

Bemerkungen

1. Um terminologische Verwechselung vermeinden zu können, sei darauf hingewiesen, daß bei $(l_1, l_2, \ldots, l_r)$-Multilevel-Zugriffsstrukturen keine Gruppenbildung der Wächter erforderlich ist. Die Zuordnung der Wächter zu einem Level führt implizit zu einer solchen Gruppebildung unter den Wächtern. Mit anderen Worten, es ist nicht gemeint, daß zunächst die Menge der Wächter in verschiedene Gruppen aufgeteilt werden und dann den Gruppenmitgliedern ein Level zugeordnet wird.
2. Implementierung mit mehr als zwei Levels seien ineffizient.

3.4 Compartmented-Zugriffsstruktur

Im Unterschied zu $(l_1, l_2, \ldots, l_r)$-Multilevel-Zugriffsstrukturen wird bei compatmented-Zugriffsstruturen die Menge der Wächter $W = \{W_1, \ldots, W_n\}$ in disjunkten Gruppen (*Comapartments*) $C_1, \ldots, C_u$, d.h. es muß gelten: $\forall\, u \bigcap_{j=1}^{u} C_j = \emptyset$. Eine compartmented-Zugriffsstruktur besteht aus Teilmengen von Wächtern, welche jeweils mindestens t_j Wächter von C_j $(j = 1, 2, \ldots, u)$ enthalten. Die gesamte Anzahl der Wächter in der Zugriffsstruktur ist mindestens t. In [27] fordert Simmons, daß $t = \sum_{j=1}^{u} t_j$ gilt.

Um zu veranschaulichen, wie autorisierte Teilmengen für die Rekonstruktion des Key gebildet werden und die Teilnehmer, die jeweils einer Gruppe zugeordnet sind, gegenseitig ersetzen können, ist einige Definitionen erforderlich.

Definition 3.6 (Ersetzbarkeit). Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas, $\mathcal{A}$ die Zugriffstruktur auf $\mathcal{W}$ und $W_i, W_j \in \mathcal{W}$ zwei beliebige Wächter mit $P_i \neq P_j$ für $i \neq j$.
Der Wächter P_i heißt *ersetzbar* durch den Wächter P_j, wenn folgende Aussage zutrifft:

$$\forall\, B \in \mathcal{A},\, P_i \in B \text{ und } P_j \notin B \Longrightarrow ((B \setminus \{P_i\}) \cup \{P_j\}) \in \mathcal{A}.$$

Wie ein Wächter durch den anderen innerhalb einer autorisierten Teilmenge $B \in \mathcal{A}$ ersetzt wird, zeigt die obige Definition. Wir müssen weiter formal den Begriff „Compartment“ im Zusammenhang mit der Ersetzung beschreiben. In der Tat sind Compartments die *Substitutenmengen* für einen Wächter $W_i \in \mathcal{W}$. Die Annahme,

daß es eine Zugriffsstruktur mit in den Definitionen genannten Eigenschaften gibt, ist erforderlich für die Konstruktion von compartmeted-Zugriffsstrukturen.

Definition 3.7 (Subtitutenmenge). Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas, $\mathcal{A}$ die Zugriffstruktur auf $\mathcal{W}$ und $W_i \in \mathcal{W}$. Die Compartment C_i ist definiert durch

$$C_i = \{W_i\} \cup \{W_j \in \mathcal{W} \mid \forall\, B \in \mathcal{A},\, P_i \in B \text{ und } P_j \notin B \Longrightarrow ((B \setminus \{P_i\}) \cup \{P_j\}) \in \mathcal{A}\}.$$

Diese Menge wird auch als *Substitutensmenge* für W_i bezeichnet und besteht aus den $W_j \in \mathcal{W}$, die den Wächter W_i $(i \neq j)$ ersetzen können.

Ist $\mathcal{A}$ eine Zugriffsstruktur, $A \in \mathcal{A}$ eine minimal autorisierte Teilmenge, muß geprüft werden, daß die Compartment C_i die Eigenschaft in der folgenden Definition besitzt.

Definition 3.8. Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas, $\mathcal{A}$ die Zugriffstruktur auf $\mathcal{W}$ und C_i. Weiter sei A eine minimal autorisierte Teilmenge.
Falls $(A \setminus C_i)$ „nicht mehr“ autorisiert ist, so sagt man C_i *partizipiert* in A.

Nun kann die Zugriffsstruktur $\mathcal{A}$ eines Secret Sharing Schemas durch Compartments $C_1, \ldots, C_u$ in $\mathcal{W}$ erweitert werden (z.B. durch (t, t)-Threshold-Zugriffsstruktur). Formal heißt es:

Definition 3.9 (Compartmented-Zugriffsstruktur). Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas, $\mathcal{A}$ die Zugriffstruktur auf $\mathcal{W}$. Die Menge $\mathcal{W}$ sei in u Compartments $C_1, \ldots, C_u$ aufgeteilt.
Für jede autorisierte Teilmenge $A \in \mathcal{A}$ Setze:

$$\mathbf{Idx}(A) = \{i \in \{1, 2, \ldots, u\} \mid C_i \textit{ partizipiert in } A\}.$$

Dann heißt die Menge , $= \{\mathbf{Idx}(A) \mid A \in \mathcal{A}\}$ eine *compartmented-Zugriffsstruktur* auf $\mathcal{A}$.

In der **3.9 Definition** enthält , keine Teilmengen von $\mathcal{W}$ und daher keine Zugriffsstruktur im Sinne der eingeführte Definition von Zugriffsstrukturen. In , werden nur die Indizes der jeweiligen Compartments C_i, die in minimal autorisierten Teilmengen partizipieren. Weiter erhebt sich die Fragen, wieviele und welche Wächter aus den Compartments in solchen minimal autorisierten Teilmengen partizipieren. Die Antwort auf die Fragen gibt die Definition

Definition 3.10 (Fixed Compartment-Threshold). Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern eines Secret Sharing Schemas, $\mathcal{A}$ die Zugriffstruktur auf $\mathcal{W}$. $C_1, \ldots, C_u$ seien Compartments in $\mathcal{W}$.
Eine ganze Zahl $t_i > 0$ heißt *fixed Compartment-Threshold* von der Compartment C_i, wenn für jede minimal autorisierte Teilmenge $A \in \mathcal{A}$, der C_i partizipiert, gilt: $|C_i \cap A| = t_i$.

Kapitel 4

Secret Sharing Schemes

4.1 Theoretische Grundlagen

Im **3. Kapitel** haben wir einige Zugriffsstrukturen vorgestellt. Die Bedeutungen von Zugriffsstrukturen zeigen, welche Teilmengen in der gegebenen Menge $\mathcal{W} = \{W_1, \ldots, W_n\}$ autorisiert sind, den Schlüssel K rekonstruieren können.

In diesem Kapitel untersuchen wir Secret Sharing Schemes für diese Zugriffsstrukturen. Liegt ein Secret Sharing Schema für eine bestimmte Zugriffsstruktur vor, so muß es möglich sein, daß

- der Dealer mit dem Schema den zu rekonstruierende Schlüssel in Shares zerlegen kann (*Share-Erzeugung*) und anschließend an die n Wächter verteilt (*Share-Verteilungsphase*). In der Share-Verteilungsphase werden die Shares aus dem Schlüssel so berechnet, daß die Rekonstruierung des Schlüssels eindeutig aus den Shares sichergestellt werden.
- es ein Rechnungsschema, teils des Secret sharing Schemmas, gibt, mit dem die autorisierten Teilmengen in der gegebenen Zugriffsstruktur mit ihren Shares den Schlüssel rekonstruieren können (*Schlüssel-Rekonstruktionsphase*).

Im allgemeinen muß ein „traditionelles“ Secret Sharing Schema mindestens die folgenden Anforderungen erfüllen:

1. Wie viele Shares erhält jeder Wächter für die Rekonstruktion, d.h. das Schema muß sein charakterisches Merkmal aufweisen (*perfekt, ideal*) sein.
2. Das Schema muß die gegebene Zugriffsstruktur realisieren, d.h. nur die autorisierten Teilmengen der n Wächter dürfen den Schlüssel mit ihren Shares rekonstruieren.
3. Welche Sicherheit vertritt das Schema (*unconditional secure, computetational secure*), d.h. wie sicher ist das Schema, wenn nicht-autorisierte Teilmengen, da

in jedem Secret Sharing Schema es konkurierende Multi-Parteien gibt, oder Angreifer den Schlüssel zu rekonstruieren. Das bedeutet aber auch, ob das Schema sicher gegen kryptographische Angriffe, die im **2. Kapitel** diskutiert sind ist.

4. Effizienz, die durch *Informationsrate* des Schemas chakterisiert ist (Bedarf an Speicherplatz) und durch festlegung des Schlüsselssraumes und die darin verwendeten Rechenoperationen (Zeitbedarf)

Diese in jedem traditionellen Secret Sharing Schema enthaltenen Anforderungen sind voneinander abhängig und üben sogar Wechselwirkungen aufeinander aus. Das Wachsen der verwendeten Shares vermindert beispielsweise einerseits die Sicherheit und erhöht andererseits die Komplexität des Schemas. Sind die Wertebereiche für den Schlüssel und die Shares identisch gewählt, so hat man ein Schema zu suchen (*ideales Schema*), das die Informationsrate 1 hat. Daraus resultiert ein speicherplatz-optimales Secret Scharing Verfahren.

Bevor wir in obige Charakristika formal im Abschnitt 4.1.1 diskutieren. Bezüglich ihrer Sicherheit klassifizieren sich Secret Sharing Schemes in zwei Kategorien: *uneingeschränkte* und *berechnungssichere* Secret Sharing Schemes. Uneingeschränkt sichere Secret Sharing Schemes werden in der Literatur *perfekte* Secret Sharing Schemes genannt, denn ihre Sicherheit der *perfekten Sicherheit* des im citebook2 eingeführten Kryptosystems . Die Sicherheit dieser Schemata werden mittels Wahrscheinlichkeits- bzw. Informationstheorie begründet. Für eine Zugriffsstruktur, die durch perfekte Secret Sharing Schemes realisieren, erhalten unautorisierten Teilmengen absolut keine Information über den Schlüssel K. Die Geheimhaltung des Schlüssels wird sogar gewährleistet, wenn die unautorisierten Teilmengen unbeschränkte Rechnenkapazität verfügen würden.
Berechnungssichere Secret Sharing Schemes basieren darauf, daß unautorisierten Teilmengen den Schlüssel rekonstruieren so schwer wie sie z.B. die große Zahl in Primfaktoren zerlegen oder die Berechnugn diskreter Logarithmen durchführen. Diese Schemata tauchen aber oft in realistischen Anwendungen auf. Wir werden sie am Ende des Anschnittes behandelt, da sie wichtig für das Konzept für die Realisierung des Schlüsselsspeicher im **Kapitel 5** benötigt werden.

Traditionelle Secret Sharing Schemes setzen stets voraus, daß die Shares auf die Wächter *geheim* verteilt werden, da die Shares private Partialinformationen sind, welche die Wächter jeweils merken bzw. geheimhalten müssen. Diese Anforderung ist allerdings für praktische Anwendungen zu streng und kann nur unter Zuhilfenahme weiterer kryptographischen Primitiven, z.B. Verschlüsselung bezw. digitalen Signatur-Schamata, implementiert werden. Dies erfordert nicht nur zusätzliche Kommunikationskosten, sondern auch aufwendige Rechenoperationen in der Abhängigkeit von den ausgewählten kryptographischen Hilfsmitteln. Werden zum Beispiel auf RSA basierte Verschlüsselung und Signatur für sichere und authentische Kommunikation zwischen den Kommunikationspartnern ausgewählt, so müssen aufwendige exponentielle Rechenoperationen modulo einer Primzahl dazu gerechnet werden.

Sollte es uns gelungen sein, ein ideales Secret Sharing Schema für die von uns aus-

gewählte Zugriffsstruktur zu finden, dann bedeutet nicht, daß es um eine speicherplatzeffizientes Schema handelt, bei dem jeder Wächter einen Share für die Rekonstruktion erhalten kann. Denn, wie erwähnt, die Wächter in einem *verifiable Secret Sharing Schema* (VSS) müssen beispielsweise neben ihrem Share zusätzliche Information vom Dealer erhalte, um zu überprüfen, ob sein Share korrekt ist oder ob der gesendete Share wirklich vom Dealer stammt. In [24] benutzt T.P. Pedersen das Shamirs (k, n)-Threshold Schema, also mit *Informationsrate* $\rho = 1$. In der Tat ist die Informationsrate seines Schemas $\rho_{VSS} = \frac{1}{2}$. Wäre stattdessen die Methode von M. Ito A. Saito and T. Nishiseki [15] für dieselbe (k, n)-Threshold-zugriffsstruktur gewählt, würden die Kommunikationskosten und die Speicherung der Shares viel höher wachsen.

4.1.1 Perfekte und ideale Secret Sharing Schemes

Zuvor haben erwänht, daß beim Secret Sharing Schema autorisierte Teilmengen von Wächtern mit ihren entsprechenden Shares den Wert des Schlüssels K rekonstruieren können. Teilmengen, die nicht in einer Zugriffsstruktur $\mathcal{A}$ enthalten sind, versuchen unter bestimmten Situationen, z.B. in Angriffsfällen, den Schlüssel K zu berechnen. Secrect Sharing Schemata, die diese Situationen ausschließen, werden in diesem Abschnitt behandelt. Unter Zuhilfenahme von Wahrscheinlichkeitsverteilung, insbesondere Entropie beschreiben wir diese für die Arbeit wichtige Klasse von Secrect Sharing Schemata.

Definition 4.1 (perfektes Secret Sharing Schema). Sei $\mathcal{A}$ eine Zugriffsstruktur auf einer Menge $\mathcal{W}$ von Wächtern. Ein Secret Sharing Schema heißt *perfekt*, wenn folgende Bedingungen erfüllt sind:

1. Jede autorisierte Teilmenge von Wächtern kann den Schlüssel mit ihren Shares rekonstruieren.
2. Nicht autorisierte Teilmengen *haben absolut keine Informationen* über den Schlüssel.

Der Ausdruck "haben absolut keine Informationen„ stellt die uneingeschänkte Sicherheit (unconditional Security) des Secret Sharing Schemas dar. Praktisch berechenbare Sicherheit haben wir im Kapitel 2 eingeführt.

Unter Zuhilfenahme von Wahrscheinlichkeitsverteilung bzw. Entropie formulieren wir die beiden Bedingungen mathematisch.

Dazu sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ eine Menge von Wächtern. $\mathcal{A}$ sei eine monotone Zugriffsstruktur auf $\mathcal{W}$. $\mathcal{K}$ sei der Schlüsselraum und $\{p(K)\}_{K\in\mathcal{K}}$ eine Wahrscheinlichkeitsverteilung auf $\mathcal{K}$. Die Menge der möglichen Shares, die der Wächter W_i bezüglich der Zugriffsstruktur $\mathcal{A}$ empfängt, werde V_i bezeichnet. bezeichne V_A die Menge $V_{i_1} \times \cdots \times V_{i_{|A|}}$. Dann wird durch das Secret Sharing Schema für Schlüssel aus $\mathcal{K}$ und der Wahrscheinlichkeitsverteilung $\{p(K)\}_{K\in\mathcal{K}}$ auf $\mathcal{K}$ eine Wahrscheinlichkeitsverteilung $\{p(v)\}_{v\in V_A}$ auf V_A induziert, wobei $v := (v_{i_1}, \ldots, v_{i_{|A|}}) \in V_A$ zu setzen ist. Die oben formulierten Bedigungen können wie folgt beschreiben:

1. Falls $A \in \mathcal{A}$ ist, dann exsistiert für alle $v \in V_A$ einen eindeutigen $K \in \mathcal{K}$, so daß $p(K|v) = 1$ gilt.

2. Falls $A \notin \mathcal{A}$, dann für alle $K \in \mathcal{K}$ und $v \in A$ gilt $p(K|v) = p(K)$ ([30], Definition 2.2, Seite 48).

Ist nun $H(\mathcal{K})$ die Entropie von $\{p(K)\}_{K\in\mathcal{K}}$ und $H(V_A)$ die Entropie von $\{p(v)\}_{v\in V_A}$, so sehen die beiden Bedingungen im informationstheoretischen Sinne so aus:

1. Für alle $A \in \mathcal{A}$ gilt $H(\mathcal{K}|V_A) = 0$.

2. Für alle $A \notin \mathcal{A}$ gilt $H(\mathcal{K}|V_A) = H(\mathcal{K})$.

In Kapitel 3 haben wir die Threshold-Zugiffsstruktur

$$\mathcal{A} = \{B \subseteq \mathcal{W} : \ |B| \geq k \text{ mit } 1 \leq k \leq n\}$$

eingeführt. Ein secret Sharing Schema, welches diese Zugriffsstruktur realisiert, heißt (k, n)-*Threshold Schema.*

Basiert auf den zueinander äquivalen obigen Definitionen kann man das (k, n)-Threshold Schema folgendes definieren [5]:

Definition 4.2 (Perfektes (k, n)-Threshold Schema). Es sei $\mathcal{A}$ eine (k, n)-Threshold-Zugriffsstruktur auf der $\mathcal{W} = \{W_1, \ldots, W_n\}$ gegeben. $\mathcal{K}$ bezeichne wieder die Menge aller möglichen Werte des Schlüssels und V_i die Menge aller moglichen Werte der Shares des Wächters W_i. Ferner bezeichne V die Menge $V_1 \times \cdots \times V_n$. Dann wird durch das Secret Sharing Schema für Schlüssel aus $\mathcal{K}$ und der Wahrscheinlichkeitsverteilung $\{p(K)\}_{K\in\mathcal{K}}$ auf $\mathcal{K}$ eine Wahrscheinlichkeitsverteilung $\{p(v)\}_{v\in V}$ auf V induziert, wobei $v := (v_1, \ldots, v_n) \in V$ zu setzen ist.

Ein (k, n)-*Threshold Secret Sharing Schema* ist eine Methode, mit welcher der Schlüssel K unter n Wächtern so geteilt wird, so daß für eine beliebige autorisierte Teilmenge A aus der Threshold-Zugriffsstruktur mit

$$A = \{W_{i_1}, W_{i_2}, \ldots, W_{i_k}\} \subseteq \mathcal{W} \text{ mit } i_1 < i_2 < \cdots < i_k.$$

die beiden Bedingungen erfüllt sind:

1. Für eine beliebige k-Tupel der Indizes $i_1, i_2, \ldots, i_k$ mit $1 \leq i_j \leq n$ und für jedes $(v_{i_1}, v_{i_2}, \ldots, v_{i_k}) \in V_{i_1} \times \cdots \times V_{i_k}$ mit $p(v_{i_1}, v_{i_2}, \ldots, v_{i_k}) > 0$, wo bei p der Wahrscheinlichkeitsmaß aus der Definition 2.1 (Seite 17) ist. Dann existiert einen eindeutigen Schlüssel K, so daß gilt:

$$p\big(K|v = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})\big) = 1.$$

2. Für ein beliebiges $j < k$, eine beliebige j-Tupel der Indizes $i_1, i_2, \ldots, i_j$ mit $1 \leq i_j \leq n$ und jedes $(v_1, v_2, \ldots, v_j) \in V_{i_1} \times \cdots \times V_{i_k}$ mit $p(v_{i_1}, v_{i_2}, \ldots, v_{i_j}) > 0$, p der Wahrscheinlichkeitsmaß wie oben. Dann gilt für jedes $K \in \mathcal{K}$ die Gleichung

$$p\big(K|v = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})\big) = p(K).$$

Unter der Benutzung der Definitionen in [7, 29, 4, 3, 16] können die beide Bedingungen 1 und 2 in der Definition 4.2 mit Hilfe der Informationstheorie formulieren.

1. Für eine beliebige k-Tupel der Indizes $i_1, i_2, \ldots, i_k$ mit $1 \leq i_j \leq n$ gilt:

$$H(K|V_{i_1}, \ldots, V_{i_k}) = 0.$$

2. Für ein beliebiges $j < k$, eine beliebige j-Tupel der Indizes $i_1, i_2, \ldots, i_j$ mit $1 \leq i_j \leq n$ gilt:

$$H(K|V_{i_1}, \ldots, V_{i_j}) = H(K).$$

Diese Definition liefert eine wichtige Tatsache, daß das (k, n)-Threshold Schema im allgeminen ein perfektes Schema ist. Ein Wächter kann damit für die Rekonstruktion des Schlüssel K mehrere Shares erhalten.

In einem perfekten Secret Sharing Schema ist $|V_i|$ eine sehr wichtige Größe für jede seiner Implementierungen. Sie ist die Mächtigkeit der Wertemange, in der das Share eines Wächter annehmen kann. Mit Hilfe der Entropie, kann man die binärspeicherung eines Share im Verhältniss zu dem Schlüssel K umrechnen.
Im allgemeinen gilt für jedes perfektes Secret Sharing Schema (PSS)

$$|V_i| \geq |\mathcal{K}| \text{ für } 1 \leq i \leq n. \tag{4.1}$$

In der angegebenen Literatur wurde gezeigt, daß die beste untere Schranke für $|V_i|$ in einem PSS bezüglich spezieller Zugriffstrukturen mit vier Wächtern [4], zum Beispiel anhand der Zugriffsstruktur $\mathcal{A} = \big\{\{W_1, W_2\}, \{W_2, W_3\}, \{W_3, W_4\}\big\}$,

$$|V_i| \geq 1.5 \cdot |\mathcal{K}| \text{ ist.} \tag{4.2}$$

Dies bedeutet, daß die Länge eines Share, das jeder Wächter beim Verteilungsprozess erhält, mindestens 50% größer ist als die Länge des Schlüssels K.

In [7] gibt L. Czirimaz eine schärfere Schranke mit Hilfe *Polymatroid-Struktur (polymatroid structure)* für $|V_i|$ an:

$$|V_i| \geq \frac{n}{\log n} \cdot |\mathcal{K}|, \tag{4.3}$$

wobei n die Anzahl der Wächter in einem PSS ist. Diese Tatsache wird das folgende Theorem (*Theorem* 7 in [7]) bestätigt:

Theorem 4.1. Für jedes n existiert eine Zugriffsstruktur $\mathcal{A}$ auf $\mathcal{W} = \{W_1, \ldots, W_n\}$. In einem beliebigen perfektens Secret Sharing Schema, welches die Zugriffsstruktur $\mathcal{A}$ realisiert, ist die Länge des Share eines Wächters mindestens $\frac{n}{\log n} \cdot |\mathcal{K}|$, wobei $\mathcal{K}$ der Schlüsselraum ist.

Die in dem Lemma verwendete Zugriffsstruktur wird als *Sperner System* bezeichnet. Dies ist eine monotone Zugriffsstruktur, in der maximale Anzahl von minimal autorisierten Teilmengen zum ersten Mal vom E. Sperner bestimmt wird [28].
Die Anwendung des Lemmas kann wie folgt beschrieben werden: Wähle eine grrößte ganze Zahl s mit $2^s + s \Leftrightarrow 2 \leq n$. Sei K der Schlüssel, bestehend aus einer Bitfolge. Die Zugriffsstruktur $\mathcal{A}$ kann nach E. Sperner als $\mathcal{A} = \{U_i | \, i = 0, 1, \ldots, 2^s \Leftrightarrow 2\}$ konstruiert werden. Für diese Zufriffsstruktur kann man ein PSS konstruieren, in dem für jedes *geheimes* Bit die Länge des Share eines Wächters höchstens n Bits ist.
Nun sei s ein geheimes Bit aus der Bitfolge. Für jedes i wähle $|U_i|$ Bits, so daß ihre Summe *modulo* 2 geleich s ist. In der Verteilungsphase werden diese Bits an die Wächter der minimal autorisierten Teilmenge U_i verteilt. Da in einer minimal monotonen Zugriffsstruktur kann ein Wächter in mehreren Teilmengen vorhanden sein, erhält jeder Wächter so viele Bits wie ihre Beteiligung. Die Länge jedes Share ist daher höchsten $2^s \Leftrightarrow 2 \leq n$.

4.1.2 Informationsrate (Information rate) eines PSS

In diesem Abschnitt wird das Verhältnis zwischen Speicherung der Shares und dem Schlüssel K in einem PSS erklärt.

Der Schlüsselsraum $\mathcal{K}$ kann als Zufallvariable betrachtet werden. Sei $\{p(K)\}_{K \in \mathcal{K}}$ eine beliebige Wahrscheinlichkeitsverteilung auf $\mathcal{K}$. Die Entrope von $\mathcal{K}$ wird im Kapitel 2 als

$$H(\mathcal{K}) = \sum_{K \in \mathcal{K}} p(K) \cdot \log p(K) \tag{4.4}$$

definiert. In diesem Zusammenhang bedeutet $H(\mathcal{K})$ die Ungewißheit dafür, daß der Schlüssel aus dem Schlüsselraum $\mathcal{K}$ bezüglich der Wahrscheinlichkeitsverteilung $\{p(K)\}_{K \in \mathcal{K}}$ gewählt wird. Für die Entropie von $\mathcal{K}$ gilt die Ungleichung

$$0 \leq H(\mathcal{K}) \leq \log |\mathcal{K}|, \tag{4.5}$$

wobei $H(K) = 0 \Leftrightarrow \exists K \in \mathcal{K} : \, p(K) = 1$ und $H(\mathcal{K}) = \log |\mathcal{K}| \Leftrightarrow \forall K \in \mathcal{K} p(K) = 1/|\mathcal{K}|$. Das ganze gilt analog für V_i.

Da $K \in \mathcal{K}$ ist, kann der Schlüssel K als eine Bitfolge der Länge, bezüglich der Binärspeicherung, $\log |\mathcal{K}|$. Auf der gleichen Weise beträgt die Länge eines Share, das der Wächter W_i als Teilinformation im PSS erhält, $\log |V_i|$ Bits.
Anhand der Ungleichung 4.1 gilt weiter

$$\log |V_i| \geq \log |\mathcal{K}| \text{ für } 1 \leq i \leq n. \tag{4.6}$$

Basierend auf diesen Resultaten wird die Informationsrate für den Wächter W_i wie folgt definiert:

Definition 4.3 (Informationsrate (information rate)). Es sei ein perfektes Seeret Sharing Schema mit Zugriffsstruktur $\mathcal{A}$ auf der Teilnehmermenge $\mathcal{P} = \{W_1, \ldots, W_n\}$

gegeben. $\mathcal{K}$ bezeichne wieder die Menge aller moglichen Werte, die der Schlüssel K bezüglich der Gleichverteilung mit $p(K) = 1/|\mathcal{K}|$ und V_i die Menge aller möglichen Werte, die ein Share des Wächters W_i annimmt. Die *Informationsrate* für den Wächter W_i ist definiert durch

$$\rho = \frac{\log |\mathcal{K}|}{\log |V_i|} \text{ für } 1 \leq i \leq n.$$

Die Informationsrate des perfekten Secret Sharing Schemas wird im allgemeinen definiert als

$$\rho = \frac{H(\mathcal{K})}{\max\limits_{W_i \in \mathcal{W}} \{H(V_i) : i = 1, 2, \ldots, n\}}. \tag{4.7}$$

Es ist gleichbedeutend wie

$$\rho = \frac{\log |\mathcal{K}|}{\max\limits_{1 \leq i \leq n} \{\log |V_i| : i = 1, 2, \ldots, n\}}. \tag{4.8}$$

Die Informationsrate ergibt sich also aus dem Quotienten von der Anzahl der Bits, die zur Speicherung des Schlüssels durch die Anzahl der Bits, die zur Speicherung der Shares benötigt werden.

Ein nützliche Interpretation für die Speicherung bezüglich der Anzah der vorgegebenen Wächter ist die *durchschnittliche Informationsrate (average information rate) des PSS*. Sie ergibt sich aus dem harmonischen Mittel der Kehrwerte der jeweiligen Informationsrate für den Wächter W_i. Es gilt:

$$\tilde{\rho} = \frac{n}{\sum\limits_{i=1}^{n} \frac{1}{\rho_i}} = \frac{n \cdot \log |\mathcal{K}|}{\sum\limits_{i=1}^{n} \log |V_i|}. \tag{4.9}$$

Im Zusammenhang mit der Entropie werden die Informationsraten ρ_i und $\tilde{\rho}$ aus der Ungleichung 4.1 hergeleitet. Wegen Ungleichung 4.2 ist $\rho = 2/3$ die „beste“ Informationsrate und aus 4.3 ist $\rho \leq \frac{n}{\log n} \cdot |\mathcal{K}|$eine schärfere Schranke für die beste Informationsrate .

In einem beliebigen secret Sharing Schema kann man zeigen, daß die folgende Ungleichung gilt:

$$\rho \leq \tilde{\rho} \leq 1. \tag{4.10}$$

Für ein perfektes Secret Sharing Schema gilt folgender

Satz 4.1. In jedem perfekten Secret Sharing Schema, das eine gegebene Zugriffsstruktur $\mathcal{A}$ realisiert, gilt für die Informationsrate $\rho \leq 1$

Beweis. Aus [30].

Das bedeutet, daß jeweils zur Speicherung des Share eines Wächters mindestens soviel Bits benötigt sind wie für die Speicherung des Schlüssels.

In praktischen Implementierung von PSS werden der Wertebereich des Schlüssels dem Wertebereich der Shares gleich gesetzt, zum gleich dem Restklassenkörper $\mathbb{Z}_p$, wobei p eine Primzahl ist. Die Primzahl p soll hinreichend groß gewählt werden, damit vollständige Suchen, Brute-Force-Angriffe, ausgeschlossen werden. Dieser Fall wird die folgende Definition bestätigt:

Definition 4.4 (Ideales Secret Sharing Schema). Ein Seeret Sharing Schema mit Zugriffsstruktur $\mathcal{A}$ heißt ideal, wenn für die Informationsrate $\rho = 1$ gilt.

Die Definition besagt lediglich, daß sowohl die Werte des Schlüssels K bzw. der Shares aus demselben Wertebereich stammen. Ob jeder Wächter ein Share in einem idealen Secret Sharing Schema erhält, hängt es von dessen Realisierungen ab (Zum Beispiel erhält jeder Wächter ein Share in (k, n)-threshold-Schema von Shamir ein Share).

Satz 4.2. Es seien $\mathcal{W} = \{P_1, \ldots, P_n\}$ eine Menge von Wächtern und $\mathcal{A}$ eine compartmented-Zugriffsstruktur auf $\mathcal{W}$. Dann gibt es ein *ideales Secret Sharing Schema* über $GF(p)$, wenn $p > \binom{n}{t}$

4.2 Implementierung von Secret Sharing Schemes

4.2.1 Das (k, n)-Threshold-Schema (A. Shamir)

A. Shamir implementiert die Threshold-Zufriffsstruktur durch ein ideales Secret Sharing Schema. Bei diesem Schema ist der Restklassenkörper $\mathbb{Z}_p$ als Wertebereich für den Schlüssel K und die Shares S_i $(1 \leq i \leq n)$. Die Realisierung erfolgt durch Polynominterpolation in $\mathbb{Z}_p$, p ist eine Primzahl.

Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ die Wächtermenge, $\mathcal{A} = \{B \subseteq : |B| \geq k,\ 1 \leq k \leq n\}$ ist die Threshold-Zugriffsstruktur auf $\mathcal{W}$. Die Konstruktion des (k, n)-Threshold-Schemas besteht aus: *Share-Verteilung* und *Schlüsselrekonstruktion*. In der orginalen Arbeit benutzt Shamit den W2achter-Index i statt die Stützstelle x_i.

Share-Verteilung

Der Dealer wählt ein Primzahl $p \geq n + 1$ und zufällig ein Polynom f vom Grad k mit $f(x) \in \mathbb{Z}_p$, $f(x) = a_0 + a_1 x + \cdots, a_{k-1} x^{k-1}$. Es setzt voraus, daß der Koeffizient $a_{k-1} \neq 0$. Der Schlüssel sei K mit $K = a_0$.

Die Shares der Wächter werden anhand des Polynoms $f(x)$ generiert. Der Dealer wählt aus $\mathbb{Z}_p$ zufälig n paarweise verschiedene x_i mit $x_i \neq x_j$ für $i \neq j$ $(1 \leq i, j \leq n)$ und berechnet $S_i = f(x_i)$. Als Shares werden $S_i := (x_i, y_i)$ jeweils an die n Wächter W_i verteilt. Die Share-Verteilung ist damit abgeschloßen.

Schlüsselrekonstruktion

In dem (k, n)-Threshold-Schema kann der Schlüssel K durch k oder mehr als k Wächter rekonstruiert. Weniger als k Wächter erhalten keine Information über den Schlüssel. Die Rekonstruktion werden durch folgende Schritte durchgeführt:

1. Jeder der an der Rekonstruktion beteiligten Wächter W_i $(i = 1, 2, \ldots, k)$ gibt sein Share S_i an den $k \Leftrightarrow 1$ Wächter bekannt. Danach stellt jeder Wächter W_i die Gelichungen

$$G_j : \quad y_j = a_0 + a_1 x_j + \cdots, a_{k-1} x_j^{k-1} \quad j = 1, 2, \ldots, k \tag{4.11}$$

   Damit erhält jeder der k Wächter ein Gleichungssystem mit k Unbekannten $a_0, a_1, \ldots, a_{k-1}$.

2. Jeder Wächter W_i kann die Koeffizienten und damit auch den Schlüssel $K = a_0$, indem er das Gleuchungssystem beispielsweise mittels des Gaußverfahren löst.

Es sei $A = \{W_{i_1}, W_{i_2}, \ldots, W_{i_k}\}$ eine autorisierte Teilmenge aus der Threshold-Zugriffsstruktur $\mathcal{A}$. Die Wächter dieser Teilmenge wollen den Schlüssel K rekonstruieren. Das Verfahren beruht auf die Polynominterpolation.

Anhand der k Gelichungen G_i (Index-Umbenennung) ergibt sich folgende linearen Gleichungssysteme mit k Gleichungen. Alle Berechnungen laufen im $\mathbb{Z}_p$.

$$\begin{pmatrix} 1 & x_{i_1} & x_{i_1}^2 & \cdots & x_{i_1}^{k-1} \\ 1 & x_{i_2} & x_{i_2}^2 & \cdots & x_{i_2}^{k-1} \\ & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{i_k} & x_{i_k}^2 & \cdots & x_{i_k}^{k-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{pmatrix} = \begin{pmatrix} y_{i_1} \\ y_{i_2} \\ \vdots \\ y_{i_k} \end{pmatrix}$$

Bezechne $\mathbf{V}$ die in den Gleichungssystemen entstandene Matrix, so hat $\mathbf{V}$ die Form einer Vandermonde-Matrix. Die Determinante kann durch die folgende Formel berechnet werden:

$$\det \mathbf{V} = \prod_{1 \leq j < l \leq k} (x_{i_l} \Leftrightarrow x_{i_j}).$$

Da die x_i $(1 \leq i \leq k)$ paarweise verschieden sind, ist $\det \mathbf{V} \neq 0$. Das lineare Gleichungssystem ist über $\mathbb{Z}_p$ eindeutig lösbar. Dies bedeutet, daß das Polynom $f(x)$ durch die Punkte (X_{i_j}, y_{i_j}) eindeutig interpoliert. Damit kann der Schlüssel durch k Wächter W_i rekonstruiert werden, indem sie $f(0) = K$ berechnen.

Eine elegante Interpolationsmethode zur Berechnung von $f(0)$ ist die Lagrang'sche Interpolationsformel:

$$f(x) = \sum_{j=1}^{k} y_{i_j} \prod_{1 \leq l \leq k, l \neq j} \frac{x_{i_l}}{x_{i_l} \Leftrightarrow x_{i_j}}. \tag{4.12}$$

Setze als Lagrange'sche Koeffizienten:

$$L_j = \prod_{1 \leq l \leq k, l \neq j} \frac{x_{i_l}}{x_{i_l} \Leftrightarrow x_{i_j}} \tag{4.13}$$

für $1 \leq j \leq k$, ergibt sich

$$K = f(0) = \sum_{j=1}^{k} L_j \cdot y_{i_j}. \tag{4.14}$$

Aus der Formel 4.14 ist es ersichlich, daß sich der Schlüssel als Linearkombination der k Shares berechnen läßt.

Die Informationsrate des Schemas kann wegen $|\mathcal{K}| = |V_i| = |\mathbb{Z}_p|$ wie folgt berechnet werden:

$$\rho = \frac{\log |\mathcal{K}|}{\log |V_i|} = \frac{\log |\mathbb{Z}_p|}{\log |\mathbb{Z}_p|} = 1.$$

Die Sicherheit des Schemas

Im folgenden soll die Sicherheit des (k, n)-Threshold-Schema von Shamir untersucht werden. Das heißt, die Frage, ob $k \Leftrightarrow 1$ Wächter den Schlüssel rekonstruieren können.

Mit dem Schema von Shamir hat der Dealer n Shares zu einem Schlüssel K generiert und anschließend an die Wächter verteilt. Es wird angenommen, daß es Bob erneut gelungen ist, $k \Leftrightarrow 1$ Shares auszuspionieren. Um das Geheimnis aufdecken zu können, muß Bob $k \Leftrightarrow 1$ Shares $S_i = (x_i, y_i)$ das Polynom f rekonstruieren. In diesem fall kann Bob höchstens $k \Leftrightarrow 1$ Gleichungen aufstellen. Damit erhält er ein lineares Gleichungssytem mit nur $k \Leftrightarrow 1$ Gleichungen und k Unbekannten $a_0, a_1, \ldots, a_{k-1}$. Je nachdem welchen Wert Bob für welchen Koeffizienten wählt, kann der Koeffizient $a_0 = K$ jeden Wert aus $\mathbb{Z}_p$ annehmen. Also $k \Leftrightarrow 1$ Wächter erhalten keine Information über den Schlüssel.

Bob kann natürlich auch mit Lagrange'scher Interpolation versuchen, anhand der ihm bekannten $k \Leftrightarrow 1$ Stützpunkte an den Schlüssel gelangen. Allerdings bemnötigt der Algorithmus für Lagrange'che Formel mindestens k Stützpunkte als Eingaben, um ein Polynom vom Grad $k \Leftrightarrow 1$ zu berechnen. Bob kann nicht anders tun, als einen weiteren Stützpunkt zu raten. Jeder mögliche Schlüssel K^* aus $\mathbb{Z}_p$ bestimmt als Stützwert $f^*(0) = K^*$ zusammen mit den Stützwerten eindeutig ein polynom $f^*(x)$, für das Bob einen Stützpunkt als Eingabe wählen kann, Damit jeder Wert aus dem Körper $\mathbb{Z}_p$ als möglicher Schlüssel.

Fassen wir es zusammen, so ist das (k, n)-Threshold Schema von Shamir uneingeschränkt sicher, da Bob in beiden Fällen mit bliebig viel zeit und beliebig größer rechenkapazität den Schlüssel nicht berechnen kann.

Wichtige Bemerkung für das Kapitel 5

In dieser Arbeit ist es wünschenswert, den Threshold-Parameter k so zu wählen, daß der Schlüssel K mit hoher Redundanz rekonstruiert werden soll. Dies kann erreicht werden, wenn man ein k ($1 \leq k \leq n,\ n$ fest) findet, so daß $|\mathcal{A}^0| = |\{B \in \mathcal{A} \mid |B| = k\}| = \binom{n}{k}$ maximal wird.
Dazu sei die Anzahl der Wächter n fest vorgegeben. Aus der Symmetrie des Binomialkoeffizienten $\binom{n}{k} = \binom{n}{n-k}$ betrachte $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$. Es genügt zu zeigen, daß $\binom{n}{k}$ im

Intervall $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$ monoton wachsend ist.
Es gilt:

$$\binom{n}{k} \cdot 1/\binom{n}{k \Leftrightarrow 1} = \frac{n!}{k!(n \Leftrightarrow k)!} \cdot \frac{(k \Leftrightarrow 1)!(n \Leftrightarrow k+1)!}{n!} = \frac{n \Leftrightarrow k+1}{k}.$$

Nach Voraussetzung gilt:

$$0 \leq k \leq \left\lfloor \frac{n}{2} \right\rfloor \Leftrightarrow 0 \leq 2k \leq n < n+1 \Leftrightarrow k+k < n+1 \Leftrightarrow k < n \Leftrightarrow k+1.$$

Daraus folgt: $\frac{n-k+1}{k} > 1$ und damit $\binom{n}{k} > \binom{n}{k-1}$. Für $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$ gilt schließlich:

$$\binom{n}{0} < \binom{n}{1} \cdots < \binom{n}{\lfloor \frac{n}{2} \rfloor \Leftrightarrow 1} > \binom{n}{\lfloor \frac{n}{2} \rfloor}.$$

$\binom{n}{k}$ wird für $k = \left\lfloor \frac{n}{2} \right\rfloor$ maximal.
Damit der Schlüssel K noch rekonstruiert werden kann, dürfen höchstens $n \Leftrightarrow k = \lfloor \frac{n}{2} \rfloor$ Shares korrupt werden.

Sonderfall des Schemas: Das (k,k)-Threshold-Schema.

Ein wichtiger Sonderfall des (k, n)-Threshold-Schemas ist das (k, k)-Threshold-Schema. Dieses Schema wird benutzt für die Konstruktion von allgemeine monotonen) Zugriffsstrukturen nach Ito,Saito, Nishiki [15] sowie nach Benaloh und Leichter [1].

Für die Realisierung von allgemeinen Zugriffsstrukturen ist das (k, n)-Threshold-Schema von Shamir nicht geeignet. Für die Realisierung allgemeiner Zugriffsstrukturen wird das folgende Schema, bestehend aus zwei Protokollen *share* und *recover*, die im folgenden beschrieben werden.

Das Protokoll *share*.

1. Der Dealer legt ein m fest, m ist hinreichend groß. Dann ist $\mathbb{Z}_m$ eine *abel'sche* Gruppe. Der Schlüssel sei $K \in \mathcal{K} = \mathbb{Z}_m$. Weiter wählt er zufällig $(k \Leftrightarrow 1)$ Elemente $s_1, s_2, \ldots, s_{k-1}$ aus $\mathbb{Z}_m$.

2. Dealer berechnet $s_k = K \Leftrightarrow \sum_{i=1}^{k-1} s_i \pmod{m}$.

3. Für $1 \leq i \leq k$, sendet Dealer als Shares die Werte s_i *sicher* an die Wächter W_i. Die Share-Verteilung ist damit abgeschlossen.

Die Rekonstruktion des Schlüssels K wird durch das Protokoll *recover* fest gelegt.

Das Protokolll *recover*.

1. Die Wächtergruppe $\{W_1, \ldots, W_k\}$ kann den Schlüssel K mit der Gleichung $K = \sum_{i=1}^{k} s_i \pmod{m}$ rekonstruieren.

Die Sicherheit des Schemas

Die Sicherheit des Schemas beruht darauf, daß $(k \Leftrightarrow 1)$ Wächter mit ihren Shares s_i $(i = 1, 2, \ldots, k \Leftrightarrow 1)$ keine Information über den Schlüssel K erhalten. Diese Perfekte Geheimheltung des Schlüssels ist die Folgerung der Sicherheitseigenschaft des (k, n)-Threshold-Schema von Shamir. In der Regel liefert das (k, k)-Threshold-Schema niedrige Informationsrate. Dazu betrachten wir das folgende Beispiel.

Beispiel
Es sei $\mathcal{W} = \{W_1, W_2, W_3, W_4\}$ die Wächtermenge. Sei $\mathcal{A}$ die Zugriffsstruktur auf $\mathcal{W}$ mit der Basis $\mathcal{A}^0 = \big\{\{W_1, W_2, W_4\}, \{W_1, W_3, W_4\}, \{W_2, W_3\}\big\}$. Die zu rekonstruierende Schlüssel sei K. Der Dealer verteilt die Shares auf die Wächter W_i $(i = 1, 2, 3, 4)$ anhand der in $\mathbb{Z}_m$ gewählte Elemente a_1, a_2, b_1, b_2 und c_1 wie folgt:

- Der Wächter W_1 erhält als Shares $s_{11} = a_1,\ s_{12} = b_1$,
- Der Wächter W_2 erhält als Shares $s_{21} = a_2,\ s_{22} = c_1$,
- Der Wächter W_3 erhält als Shares $s_{31} = b_2,\ s_{32} = K \Leftrightarrow c_1$ und
- Der Wächter W_3 erhält als Shares $s_{41} = K \Leftrightarrow a_1 \Leftrightarrow a_2,\ s_{32} = K \Leftrightarrow b_1 \Leftrightarrow b_2$.

Mit diesen Shares kann die autorisierte Teilmenge $\{W_1, W_2, W_4\}$ als Schlüssel $K = s_{11} + s_{21} \Leftrightarrow s_{41} \pmod m = a_1 + a_2 + (K \Leftrightarrow a_1 \Leftrightarrow a_2) \pmod m$ berechnen. Die autorisierte Teilmenge $\{W_1, W_3, W_4\}$ rekonstruiert den Schlüssel K mit $K = s_{12} + s_{31} + s_{41} \pmod m = b_1 + b_2 + (K \Leftrightarrow b_1 \Leftrightarrow b_2) \pmod m$ sowie $\{W_2, W_3\}$ mit $K = s_{22} + s_{32} \pmod m = c_1 + (K \Leftrightarrow c_1) \pmod m$.
Da in diesem Schema jeder der Wächter zwei Shares aus $\mathbb{Z}_m$ erhält, beträgt die Informationsrate des Schemas $\rho = \dfrac{\log m}{\log m^2} = \dfrac{1}{2}$. Der Beweis, daß das verwendete Schema perfekt ist, wird anhand der *maximal unautorisierten Teilmengen* von $\mathcal{W}$ ausführlich in [30] beschrieben.

4.3 Effiziente Konstruktion von Secret Sharing Schemes

Im folgenden wird das Schema von Benaloh und Leichter formal beschrieben.
Es seien $\mathcal{W} = \{W_1, \ldots, W_n\}$ die Wächtermenge und $\mathcal{A}$ die Zugriffstruktur auf $\mathcal{W}$. Die Zuordnung der Shares zu den Wächtern werden durch die Transformation $\mathcal{T}$ wie folgt definiert:

R1. $\mathcal{T}(S_x, V_i)$ ordnet dem Wächter W_i das Share S_x zu.

R2. $\mathcal{T}(S_x, \vee(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathcal{T}(S_x, A_i)$.

R3. $\mathcal{T}(S_x, \wedge(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathcal{T}(S_i, A_i)$, wobei die Werte S_i mit einem (k, n)-Threshold-Schema so gewählt werden, daß $S_x = \sum_{i=1}^{n} S_i \pmod p$ gilt.

R4. $\mathcal{T}(S_x, Threshold_{(k,n)}(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathcal{T}(S_i, A_i)$, wobei die Werte S_i mit einem (k, n)-Threshold-Schema so erzeugt werden, daß daß sich der Wert S_x durch die S_i rekonstruieren läßt.

Für die **R.2** und **R.3** ergeben sich für zwei Operatoren die Sonderfälle

R2. $\mathcal{T}(S_x, A_1 \vee A_2) = \mathcal{T}(S_x, A_1) \cup \mathcal{T}(S_x, A_2)$.

R3. $\mathcal{T}(S_x, A_1 \wedge A_2) = \mathcal{T}(S_1, A_1) \cup \mathcal{T}(S_2, A_2)$ mit $S_x = S_1 + S_2 \pmod p$.

Das folgende Beispiel soll die Bedeutung der Regeln **R1-R4** verdeutlichen.

R1. Mit $\mathcal{T}(7, V_3)$ wird dem Wächter W_3 das Share $S_3 = 7$ zugeordnet.

R2. Mit $\mathcal{T}(7, A_1 \vee A_2) = \mathcal{T}(7, A_1) \cup \mathcal{T}(7, A_2)$ wird den beiden logischen Ausdrücken A_1 und A_2 jeweils der Wert 7 für weitere Berechnungen in den Regeln **R1-R4** zugeordnet.

R.3 Mit $\mathcal{T}(7, A_1 \wedge A_2) = \mathcal{T}(8, A_1) \cup \mathcal{T}(10, A_2)$ wird dem Teilausdruck A_1 der Wert 8 und dem Teilausdruck der Wert 10 für die Primzahl $p = 11$ für weitere Berechnungen in den Regeln **R1-R4** zugeordnet, da in $\mathbb{Z}_{11}$ der Wert 7 mit $7 = 8 + 10 \pmod{11}$ entsteht.

R.4 Für die Operation $Threshold_{2,3}(A_1, A_2, A_3)$, welche für die Abkürzung steht, kann das $(2, 3)$-Threshold-Schema von Shamir benutzt werden, um die entsprechende Werte für die Teilausdrücke $A_1, \ldots, A_3$ zu erzeugen. Dabei wird das Polynom $f(x) = 7 + 2x$ im $\mathbb{Z}_{11}[x]$ gewählt, so daß sich folgende Werte für $A_1, \ldots, A_3$ ergibt:

$$\begin{aligned} A_1 : &\quad S_1 = f(1) = 9 \\ A_2 : &\quad S_2 = f(2) = 0 \\ A_3 : &\quad S_3 = f(3) = 2. \end{aligned}$$

Wie andere Schema wird das Schema von Benaloh und Leichter durch zwei Protokolle beschrieben, indem die Shares zu dem Schlüssel K berechnet und anschließend auf die Wächter W_i verteilt sowie die Rekonstruktion des Schlüssels durch die minimal autorisierten Teilmengen der Zugriffstruktur $\mathcal{A}$, oder mit anderen Worten durch die Basis $\mathcal{A}^0$ von $\mathcal{A}$, rekonstruiert.

Das Protokol *share*

1. Vorgelegt sei die Zugriffsstruktur $\mathcal{A} = \{B_1, \ldots, B_r\}$ mit B_ξ $(1 \leq \xi \leq r)$ als autorisierten Teilmengen, die berechtigt sind, den Schlüssel K zu rekonstruieren. Dann bestimmt Dealer eine Primzahl $p > K$, so daß die Regeln **R3** und **R4** im Restklassenkörper $\mathbb{Z}_p$ gültig sind.

2. Der Dealer findet die Basis $\mathcal{A}^0$ der Zugriffsstruktur $\mathcal{A}$.

3. Dann transformiert Dealer die Basis $\mathcal{A}^0 = \{ \{W_{1,1}, \ldots, W_{1,m}\}, \ldots, \{W_{r,1}, \ldots, W_{r,t}\} \}$ in die logische Formel A der Form $A = \big((V_{1,1} \wedge \cdots \wedge V_{1,m}) \vee \cdots \vee (V_{r,1} \wedge \cdots \wedge V_{r,t})\big)$. Mit dem Methode der mathematischen Logik (Literatur) formt Dealer die logische Formel A zu einer logisch äquivalenten A_{min} um, in der die Anzahl der Variablen minimal wird.

4. Die Shares für die Wächter W_i berechnet der Dealer, indem er die eingeführten Regeln anhand der Klammerungshierarchie rekursiv auf die Formel A_{min} anwendet. Beginnend mit $\mathcal{T}(K, A_{min})$ erhält der Dealer bei jeder Anwendung von $\mathcal{T}(S_{i,j}, V_i)$ jeweils ein Share für den Wächter W_i.

5. Um die Verteilungsphase abzuschließen, veröffentlicht Dealer die Zugriffstruktur $\mathcal{A}$, die Formel A_{min} und die Primzahl p und sendet an jeden Wächter W_i die Shares $S_{i,j}$ in der Reihenfolge, in der sie in der Formel A_{min} aufgetreten sind.

Das Protokoll *recover*

Jeder der an der Rekonstruktion beteiligten Wächter W_i gibt seine Shares $S_{i,j}$ den anderen Wächtern bekannt. Jeder dieser Wächter kann den Schlüssel K anhand der Klammerungshierarchie „von inener nach außer“ rekonstruieren. Dabei setzt er die bekannten Shares für die entsprechenden Variablen ein, so daß die folgenden gilt:

1. Jedes bekannte Share $S_{i,j}$ ersetzt die logische Variable $V_{i,j}$.

2. Jeder Ausdruck $(V_1 \vee \cdots \vee K \vee \ldots V_m)$ wird durch K ersetzt.

3. Jeder Ausdruck $(S_1 \wedge \cdots \wedge S_m)$ wird durch $K = \sum_{i=1}^{m} S_i \pmod p$ ersetzt.

4. Jeder Ausdruck $Threshold_{k,n}(S_1, \ldots, S_n)$ wird durch den Wert ersetzt, welchen das (k, n)-Threshold-Schema mit den Shares $S_1, \ldots, S_n$ als Ergebnis liefert.

Schließlich ergibt sich der Schlüssel als der Wert, mit dem die Formel A_{min} ersetzt wird. Zur Rekonstruktion werden die Shares, welche mit den Regeln **R1-R4** aus der Formel A_{min} entstanden sind, wieder in diese Formel eingesetzt und die mit den Regeln **R1-R4** zusammenhängenden Operation umgekehrt. Auf diese Weise kann der Schlüssel K rekonstruiert werden.
Da die Zugriffsstrukturen und tranformierten logischen Formeln äquivalent sind, können die Wächter jeder autorisierten Teilmengen von der Zugriffsstruktur $\mathcal{A}$ mit ihren Shares, welche durch die Anwendung der Regeln **R1-R4** entstanden sind, den Schlüssel rekonstruieren.

Die Sicherheit des Schemas

Nun wird gezeigt, daß das Schema von Benaloh und Leichter perfekt ist. Dazu sei $N = \{W_1, \ldots, W_h\} \notin \mathcal{A}$. Die Shares der Wächter zu dem Schlüssel K und für die logische Formel A sei vom Dealer anhand der Zuordnung $\mathcal{T}(K, A)$ bestimmt. Weiter soll gelten, daß sich jede logische Formel A mit $d > 0$ Operatoren als $A = \mathcal{O}(A_1, \ldots, A_n)$ darstellen läßt, wobei sich um die Operatoren $\mathcal{O} \in \{\vee, \wedge, Threshold_{(k,n)}\}$ und bei den A_i um die logische Ausdrücke mit weniger als d Operatoren handelt. Dann kann die Sicherheit des Schema durch Induktion über die Anzahl der Operatoren in A bewiesen werden. Um Information über den Schlüssel zu erhalten, sind die Wächter $\{W_1, \ldots, W_h\}$ auf die Shares angewiesen, die sie für jeden Ausdruck A_j erhalten haben. Dabei wird mit $share(W_i, A_j)$ die Menge die Menge der Shares bezeichnet, die der Wächter W_i für den Ausdruck A_j erhält, so daß die Menge der Shares, die den Wächtern $\{W_1, \ldots, W_k\}$ zu einem Ausdruck A_j bekannt sind, mit $share(A_j) = \left\{ \bigcup_{i=1}^{k} share(W_i, A_j) \right\}$ gegeben ist.

Induktionsanfang
Betrachtet wird nun ein Ausdruck $A = (V_i)$, der nur aus der Variable V_i besteht. Damit besteht die monotone Zugriffsstruktur $\mathcal{A}$ aus allen Wächtergruppen, in denen die Wächter W_i partizipiert ist. Das Schema ordnet dann jedem Wächter W_i dem eigentlichen Schlüssel K als Share zu, während ansonstens keine weiteren Shares verteilt werden. Somit erhält eine Gruppe von Wächtern, in der der Wächter W_i nicht patizipiert ist, keine Information über den Schlüssel K.

Induktionsannahme
Die Induktionsannahme für den Ausdruck $A = \mathcal{O}(A_1, \ldots, A_n)$ besteht darin, daß die Shares, die den Wächtern $\{W_1, \ldots, W_h\}$ zu dem Teilausdruck A_j bekannt sind, keine Information über den Schlüssel offenbaren, d.h. es lassen sich aus der Menge $share\ (A)$ keine Information über den Schlüssel ableiten.

Induktionsschluß
Es muß noch gezeigt werden, daß sich auch aus der Kommbination aller Shares der Wächter $W_1, \ldots, W_h$ keine Information über den Schlüssel ableiten läßt. Dazu betrachten wir die drei Operation $\vee$, $\wedge$ und $Threshold_{(k,n)}$ in dem Ausdruck $\mathcal{O}(A_1, \ldots, A_n)$.

Zunächst handelt es sich um den Operator $\vee$, womit der Ausdruck der Form $A = \vee(A_1, \ldots, A_n)$ annimmt. Die Shares zu diesem Ausdruck werden über die Zuordnung

$$\mathcal{T}(K, A) = \mathcal{T}(K, \vee(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathcal{T}(K, A_i)$$

erstellt.
Die Shares für zwei verschiedene Teilausdrücke A_i, A_j $(i \neq j)$ werden die über die Zuordnung $\mathcal{T}(K, A_i)$, $\mathcal{T}(K, A_j)$ unabhängig von einander erstellt, d.h. die Werte der Shares für den Teilausdrück A_i hängen nicht mit den Werten der Shares für Teilausdruck A_j zusammen. Damit ergibt sich auch die Kommbination der Shares, die

den Wächtern $\{W_1, \ldots, W_h\}$ für verschiedene Teilausdrücke bekannt sind, keine Information über das Geheimnis K. Nach der Induktionsannahme $\{W_1, \ldots, W_h\}$ aus den Shares für jeden einzelnen Teilausdruck A_j ebenfalls keine Information über den Schlüssel K ableiten. Da die Wächter nur von ihren Shares auf den Schlüssel schließen können, ergibt sich insgesamt, daß sie keine Information über den Schlüssel erhalten können.

Nun handelt es sich um den Operator $\wedge$, so daß der logische Ausdruck $A = \wedge(A_1, \ldots, A_n)$. Zu dem Ausdruck A ist die Zuordnung $\mathcal{T}$ mit

$$\mathcal{T}(K, A) = \mathcal{T}(K, \wedge(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathcal{T}(S_i, A_i)$$

gegeben, wobei $K = \sum_{i=1}^{n} S_i \pmod p$ gilt. Für die Wächter einer unautorisierten Teilmenge gibt es mindestens einen Index i, für den die Shares der Wächter nicht ausreichen, um den Wert des Schlüssels K zu berechnen. ansonsten könnte diese Wächtergruppe den Schlüssel mit ihren Shares anhand des Schemas bestimmen und würde damit die autorisierte Teilmenge bilden, was ein Widerspruch zu der gemachten Annahme wäre. Die Shares für verschiedene Teilausdrücke werden unabhängig voneinander erzeugt, so daß sich auch mit einer Kombination aller Shares, die für andere Teilausdrücke mit $\mathcal{T}(S_j, A_j)$ $i \neq j$ bestimmt werden, keine Information über S_i ableiten lassen. Die Wächter einer unautorisierten Teilmenge können daher nicht alle Werte S_i berechnen, so daß für diese Wächter $K = \sum_{i=1}^{n} \pmod p$ und damit auch der Schlüssel völlig unbestimmt bleiben.

Schließlich handelt es sich um den Operator $Threshold_{(k,n)}$ und bei dem logischen Ausdruck um $A = Threshold_{(k,n)}(A_1, \ldots, A_n)$. Die Zuordnung $\mathcal{T}$ ist dann mit

$$\mathcal{T}(K, A) = \mathcal{T}(K, Threshold_{(k,n)}) = \bigcup_{i=1}^{n} \mathcal{T}(S_i, A_i)$$

gegeben, wobei die S_i mit einem (k, n)-Threshold-Schema bestimmt werden. Nach Definition von (k, n)-Threshold-Schema erhalten die Gruppe mit weniger als k Wächtern keine Information über den Schlüssel K. Die Anzahl der Wächter jeder unautorisierten Teilmenge im Ausdruck A muß kleiner als k sein, da diese Wächter ansonsten den Schlüssel anhand des Schemas bestimmen könnten und somit als autorisierte Teilmenge in der Zugriffsstruktur $\mathcal{A}$ enthalten sein müßten, Zusammen mit der Unabhängigkeit der Shares, die für verschiedene Teilausdrücke erzeugt werden, folgt auch in diesem Fall, daß die Teilmenge, die nicht in der Zugriffsstruktur $\mathcal{A}$ enthalten ist, keine Information über den Schlüssel aus ihren Shares ableiten können.

Implemetierungsbeispiel

Das folgende Beispiel zeigt, wie das Schema von Benaloh und Leichter funktioniert. Der Schlüssel $K = 3$ wird mit dem Protokoll *share* auf die Wächtermenge $\mathcal{W} = \{W_1, \ldots, W_6\}$ verteilt.

Share-Verteilung

Es sei $\mathcal{A}$ die Zugriffsstruktur auf $\mathcal{W}$. Dealer wählt eine Primzahl $p = 11$, so daß die Regel **R3** und das in **R4** verwendete (k, n)-Threshold-Schema von Shamir in $\mathbb{Z}_{11}$ gültig sind.
Weiter bestimmt Dealer die Basis $\mathcal{A}^0$ von der Zugriffsstruktur $\mathcal{A}$ anhand der Definition $\mathcal{A}^0 = \{B \in \mathcal{A} \mid \forall A \in \mathcal{A}, A \nsubseteq B\}$. Demnach ist die die Basis $\mathcal{A}^0$ gegeben durch

$$\mathcal{A}^0 = \big\{ \{W_1, W_2\}, \{W_1, W_3\}, \{W_3, W_4, W_5\}, \{W_3, W_4, W_6\}, \{W_4, W_5, W_6\} \big\}.$$

Nun wird die Basis $\mathcal{A}^0$ in den logischen Ausdruck A transformiert, wobei jedem Wächter W_i eine logische Variable V_i zugeordnet wird. Es gilt:

$$A = \big((V_1 \wedge V_2) \vee (V_1 \wedge V_3) \vee (V_3 \wedge V_4 \wedge V_5) \vee (V_3 \wedge V_4 \wedge V_6) \vee (V_4 \wedge V_5 \wedge V_6)\big).$$

Durch die Ausklammerung der beiden Variablen $V_1,\ V_4$ wird der Ausdruck A zunächst vereinfacht, so daß der Ausdruck A mit dem folgenden logisch äquivalent ist:

$$\Big(\big(V_1 \wedge \big(V_2 \vee V_3\big)\big) \vee \Big(V_4 \wedge \big((V_3 \wedge V_5) \vee (V_3 \wedge V_6) \vee (V_5 \wedge V_6)\big)\Big)\Big).$$

Der Teilausdruck $\big((V_3 \wedge V_5) \vee (V_3 \wedge V_6) \vee (V_5 \wedge V_6)\big)$ wird durch $Threshold_{(2,3)}(V_3, V_5, V_6)$, so daß Dealer als minimierten logischen Ausdruck A_{min} erhält:

$$A_{min} = \Big(\big(V_1 \wedge \big(V_2 \vee V_3\big)\big) \vee \big(V_4 \wedge Threshold_{(2,3)}(V_3, V_5, V_6)\big)\Big)$$

Nun bestimmt Dealer die Shares für die Wächter W_i, indem zunächst gesetzt werden:

$$\begin{aligned} A_1 &:= (V_2, V_3) \\ A_2 &:= Threshold_{(2,3)}(V_3, V_5, V_6) \end{aligned}$$

Die Berechnug der Shares S_i $(i = 1, 2, 3, 4, 5, 6))$ der Wächter werden dann durch die Anwendung der Regeln **R1**-**R4** aud den Ausdruck A_{min} rekursiv berechnet:

R2. $\mathcal{T}\Big(K, \big((V_1 \wedge A_1) \vee (V_4 \wedge A_2)\big)\Big) = \mathcal{T}\big(K, (V_1 \wedge A_1)\big) \cup \mathcal{T}\big(K, (V_1 \wedge A_1)\big).$

R2. $\mathcal{T}\Big(3, \big((V_1 \wedge A_1) \vee (V_4 \wedge A_2)\big)\Big) = \mathcal{T}\big(3, (V_1 \wedge A_1)\big) \cup \mathcal{T}\big(3, (V_1 \wedge A_1)\big).$

R3. $\mathcal{T}\big(3, (V_1 \wedge A_1)\big) = \mathcal{T}(5, V_1) \cup \mathcal{T}(9, A_1)$ mit $3 = 5 + 9 \pmod{11}$.

R3. $\mathcal{T}\big(3, (V_4 \wedge A_2)\big) = \mathcal{T}(10, V_4) \cup \mathcal{T}(4, A_2)$ mit $3 = 4 + 10 \pmod{11}$.

R2. $\mathcal{T}(9, A_1) = \mathcal{T}\big(9, (V_2 \vee V_3)\big) = \mathcal{T}(9, V_2) \cup \mathcal{T}(9, V_3)$.

R4. $\mathcal{T} = \mathcal{T}\big(4, Threshold_{(2,3)}(V_3, V_5, V_6)\big) = \mathcal{T}(3, V_3) \cup \mathcal{T}(6, V_5) \cup (2, V_6)$, wobei die Werte 3, 6 und 2 durch das $(2, 3)$-Threshold-Schema von Shamir in $\mathbb{Z}_{11}$ mit dem Polynom $f(x) = 4 + 7x$ gerechnet wurden.

R1. Mit $\mathcal{T}(5, V_1)$ erhält der Wächter W_1 als sein Share $S_1 = 5$. Auf gleicher Weise erhält der Wächter W_2 das Share $S_2 = 9$, W_4 das Share $S_4 = 10$, W_5 das Share $S_5 = 6$, W_6 das Share $S_6 = 2$ und der Wächter W_3 zwei Shares $W_{3,1} = 9$ und $W_{3,2} = 3$.

Um die Share-Verteilung abzuschließen, veröffentlicht Dealer die Zugriffsstruktur $\mathcal{A}$, A_{min} und die Prinzahl $p = 11$ und gibt ebenfalls bekannt, daß das $(2, 3)$-Threshold-Schema von Shamir verwendet wird. Die Shares sendet der Dealer nun an die Wächter W_i.

Schlüssel-recovery

Wollen die Wächter beispielsweise der (minimalen) autorisierten Teilmenge $\{W_4, W_3, W_5\}$ den Schlüssel K rekonstruieren, dann setzen sie ihre Shares in den Ausdruck A_{min} ein:

$$A_{min} = \big((S_1 \wedge (S_2 \vee 9)\big) \vee \Big(10 \wedge Threshold_{(2,3)}(3, 6, S_6)\big)\Big).$$

Der Ausdruck $Threshold_{(2,3)}(3, 6, S_6)$ liefert den Wert 4, welcher durch Shamirs $(2, 3)$-Threshold-Schema aus den beiden Werten 3 und 6 rekonstruiert werden, so wird der Ausdruck A_{min} vereinfacht:

$$A_{min} = \big((S_1 \wedge (S_2 \vee 9)\big) \vee \big(10 \wedge 4)\big).$$

Der Ausdruck $(10 \wedge 4)$ liefert den Wert 3, da sich $3 = 4 + 10 \pmod{11}$ ergibt.

Der Ausdruck $\Big(\big(S_1 \wedge (S_2 \vee 9)\big) \vee 3\Big)$ liefert schließlich den Wert 3. Somit wird der Schlüssel $K = 3$ rekonstruiert.

Bemerkungen

- Da W_3 zwei Shares $W_{3,1} = 9$ und $W_{3,2} = 3$ erhält, ist die Informationsrate in diesem Schema:

$$\rho = \frac{\log |\mathbb{Z}_{11}|}{\log |\mathbb{Z}_{11}|^2} = \frac{1}{2} \cdot \frac{\log |\mathbb{Z}_{11}|}{\log |\mathbb{Z}_{11}|} = \frac{1}{2}.$$

- Mit der Methode von M. Ito, A. Saito und T. Nishiseki erhählt jeder Wächter so viele Shares wie die Anzahl der autorisierten Teilmengen, in denen er partizipiert ist. Würde der Schlüssel K mit dieser Methode rekonstruiert werden, so hätte der Wächter W_3 drei Shares erhälten. Die Informationsrate wäre: $\rho_s = \frac{1}{3}$.

- Bezüglich der Threshold-Zugriffsstruktur auf $\mathcal{W} = \{W_1, \ldots, W_n\}$, welche durch das (k, n)-Threshold-Schema von A. Shamir realiziert wird, ist die Informationsrate des Schemas bekanntlich $\rho = 1$ und das Schema definitionsgemäß ideal, wobei jeder Wächter W_1 ein Share S_i erhält.
  Wird die Konstruktionsmethode von Benaloh und Leichter auf die Threshold-Zugriffsstruktur angewandt, so verkleinert sich die Informationsrate auf $\rho =$

$1/\binom{n-1}{k-1}$. Und für $1 < k < n$ ist sie noch kleiner [31] [29] [30]. Unter dieser Vorausetzungen, ist das (k, n)-Threshold-Schema nicht ideal und daher perfekt. Jeder Wächter muß mindestens ein Share erhalten.

- Mit ie Konstruktionsmethode von Benaloh und Leichter, also die Transformation von monotone Zugriffsstrukturen in disjunktive Normalformen, kann die Informationsrate höchstens $\rho = \frac{1}{2}$ erreicht werden. Mehr Informationen sind im Abschnitt „Theoretische Grundlagen von SSS“ zu entnehmen.

4.4 Anwendungsorientierte Secret Sharing Schemes

4.4.1 Das Secret Sharing Schema von Tompa und Woll

Das (k, n)-Threshold-Schema ist ein ideales Schema. Es realisiert die Threshold-Zugriffsstrukturen durch Polynominterpolation. Dieses Schema ist außerdem effizient, da jeder Wächter nur ein Share für die Rekonstruktion, so daß die Speicherung der Shares nicht intensiv und die Kommunikationskosten nicht sehr hoch ist. Dieses Schema setzt allerdings voraus, daß die Wächter gegenseitig vertrauen müssen. Diese Anforderung sind in realistischen Anwendungen zu streng, denn der Schlüssel kann durch Betrug falsch rekonstruiert werden bzw. unehrliche Wächter können andere betrügen, um allein den Schlüssel zu rekonstruieren.

In diesem Schema lösen Tommpa und Woll [32] das Problem, wie die ehrliche Wächter Betrugsversuchen mit hoher Wahrscheinlichkeit aufgedecken und den Schlüssel gegenüber einem Betrüger „verstecken“ können.

Wie das (k, n)-Threshold-Schema von Shamir legt der Dealer den Restklassekörper $\mathbb{Z}_p$ fest, wobei p eine Primzahl ist. Alle Werte für die Shares und den Schlüssel K liegen im Intervall $[0, \ldots, p]$. Um einen Betrugsversuch aufdecken zu können, wird das Intervall für ein $s \ll p$ in zwei Intervalle zerlegt, so daß die *legal* rekonstruierbare Werte $K \in [0, \ldots, s \Leftrightarrow 1]$ und *illegal* $\tilde{K} \in [s, \ldots, p]$ liegen. Wenn ein Betrüger W_j die anderen Wächter betrügen will, muß er ein gefälschtes Share S_j^* ausgeben. Zusammen mit den Shares der verbleibenden Wächter in einer autorisierten Teilmenge wird zwar einen falschen Schlüssel $\tilde{K} \neq K$ rekonstruiert, aber der Rekonstruktionsvorgang ist legal. In dem Schema von Tompa und Woll verwendet der Dealer statt die Nummer i des Wächter W_i eine zufällig gewählte Stützstelle x_i, so daß der Betrüger nicht mehr ein solches fälsches Share S_j^* berechnen und muß daher einen Wert für S_j^* raten.
In dem folgenden wird für die Sicherheit des Schemas hergeleitet, daß die Wahrscheinlichkeit für einen erfolgreichen Betrug höchstens $\frac{(s \Leftrightarrow 1)(k \Leftrightarrow 1)}{p \Leftrightarrow k}$ beträgt. Damit kann der Dealer eine beliebig kleine Wahrscheinlichkeitsschranke ε auswählen, so daß sich die Wahrscheinlichkeit für einen erfolgreichen Betrug durch geeignete Auswahl von p uns s noch kleiner als die Schranke $\varepsilon > \frac{(s \Leftrightarrow 1)(k \Leftrightarrow 1)}{p \Leftrightarrow k}$ angeben läßt.

Auf diese Weise kann einen Betrug entdeckt werden, der stattgefunden hat. Das Problem ist aber noch nicht vollständig gelöst, denn der Betrüger hat nun den Schlüssel, während die ehrlichen Wächter einen falschen rekonstruieren. Um dieses Problem zu lösen, bekommt jeder Wächter mehrere Shares, die durch einen Index in einer bestimmten Reihenfolge festgelegt wird. Die Shares mit einem bestimmten Index, den nur der Dealer kennt, rekonstruieren den eigentlichen Schlüssel K, während die mit anderem eine feste Zahl K_{dummy} bestimmen. Zur Rekonstruktion decken die beteiligten Wächter ihre Shares bezüglich der durch den Index festgelegten Reihenfolge in diskreten Runden auf und überprüft die entstandenen Werte. Wenn sich der Wert K_{dummy} ergibt, werden die Shares der nächsten Runde aufgedeckt. Ergibt sich ein illegaler Schlüssel, so haben die Wächter einen Betrugsversuch aufgedeckt und sie brechen den Rekonstruktionsvorgang ab. Wenn sich allerdings ein legaler Schlüssel ergibt, so haben die Wächter den eigentlichen Schlüssel rekonstruiert.

Die folgenden Protokolle stammen aus [32]. Mit dem Protokoll *share* erzeugt der Dealer die Shares für n Wächter und anschließenn auf sie verteilt. Die Rekonstruktion erfolgt wahlweise durch die Protokolle *recover* nach Gauß-verfahren und *recover* nach Lagrange. Zur Rekonstruktion werden mindestens k Wächter benötigt.

Das Protokoll *share*

1. Der Dealer wählt eine Zahl $s > K$ und die gewünschte Sicherheitsschranke ε. Die beiden Zahlen sind geheim. Er veröffentlicht eine Primzahl $p > \dfrac{(s \Leftrightarrow 1)(k \Leftrightarrow 1)}{\varepsilon + k}$. Die Restklassekörper sei $\mathbb{Z}_p$.

2. Dealer wählt zufällig ein Polynom f vom Grad $(k \Leftrightarrow 1)$ mit $f(x) = c_0 + c_x + \cdots + c_{k-1}x^{k-1}$, wobei $c_{k-1} \neq 0$ und $K = c_0$.

3. Der Dealer wählt n voneinader vershiedene Zahlen $x_1, \ldots, x_n \in \mathbb{Z}_p$ und berechnet die Werte $s_i = f(x_i)$.

4. Dealer sendet an jeden Wächter W_i $(i = 1, 2, \ldots, n)$ als Shares $S_i = (x_i, s_i)$.

Die Rekonstruktion kann wahlweise durch ein der beiden Protokolle erfolgt werden.

Das Protokoll *recover*

Die Implementierung dieses Protokolls basiert auf den Algorithmus zur Lösung des linearen Gleichungssystems nach Gauß. Alle Rechnungen sind *modulo* p.

1. Sei B eine autorisierte Teilmenge aus der Threshold-Zugriffstruktur $\mathcal{A} = \{B : |B| \geq k\}$. Jeder Wächter $W_j \in B$ $(j = 1, 2, \ldots, k)$ gibt als sein Share $S_j = (x_j, s_j)$ den $(k \Leftrightarrow 1)$ Wächtern bekannt. Dann rekonstruiert jeder Wächter anhand der ihm bekannten k Shares $S_j = (x_j, s_j)$ mit $s_j = f(x_j)$ die Gleichung

$$G_j : \quad s_j = c_0 + c_1 x + \cdots + c_{k-1}x^{k-1}.$$

Demnach erhält jeder Wächter ein lineares Gleichungssystem mit k Gleichungen G_j und k Unbekannten $c_0, c_1, \ldots, c_{k-1}$.

2. Jeder Wächter kann die Werte $c_0, c_1, \ldots, c_{k-1}$ und damit das Polynom vom Grad $(k \Leftrightarrow 1)$ eindeutug rekonstruieren, indem er das Gleichungssystem nach dem Verfahren von Gauß löst

3. Den Schlüssel $K = c_0$ erhält jeder Wächter durch $f(0) = c_0$.

Als Alternative kann das folgende Recover-Protokoll durchgeführt werden, welches auf der oft in der Praxis eingesetzte Lagrange'sche Interpolationsformel basiert, um den Schlüssel K zu rekonstruieren.

Das Protokoll *recover*

1. Jeder Wächter $W_j \in B$ $(j = 1, 2, \ldots, k)$ gibt als sein Share $S_j = (x_j, s_j)$ den $(k \Leftrightarrow 1)$ Wächtern bekannt.

2. Jeder Wächter extrapoliert mit den Werten als Eingabe für den Algorithmus zur Lagrange'schen Interpolation den Schlüssel K als Wert des Polynoms in $x = 0$.

Das Schema von Tompa und Woll unterscheidet sich von dem Shamirs Schema in zwei Punkte. ersten wird das Intervall $[0, \ldots, p \Leftrightarrow 1]$ durch die vom Dealer ausgewählte Zahl s in zwei Intervalle $[0, \ldots, s \Leftrightarrow 1]$ und $[s, \ldots, p \Leftrightarrow 1]$ zerlegt, so daß das eine der Wertbereich für legale und das andere der für illegale Schlüssels ist. Damit kann jeder ehrliche Wächter entdecken, daß einen Betrug stattgefunden hat, wenn einen illegalen Wert für K entstanden ist. Zweitens sind als Stützstellen die Nummer i der Wächter, mit denen sich die zugehörige Shares $s_i = f(i)$ errechnen, nicht mehr öffentlich. Der Grund dafür ist, wie erwähnt, daß jeder Betrüger, der diese Stützstellen der auf die Rekonstruktion beteiligten Wächter kennt, ein gefälschtes Share berechnen kann und damit sich ein falscher legaler Schlüssel bei der Rekonstruktion ergibt.

Um es zu verdeutlichen, betrachten wir den Fall, bei dem der Wächter W_1 neben seinem eignen Share (x_1, s_1) noch die die Stützstellen $x_2, x_3, \ldots, x_k$ der anderen $(k \Leftrightarrow 1)$ beteiligten Wächter $W_2, W_3, \ldots, W_k$ kennt. Der Betrüger kann mittels der Lagrange'schen Interpolationsformel durch die Punkte $(0, \Leftrightarrow 1), (x_2, 0), \ldots, (x_k, 0)$ das Polynom $f^*(x)$ konstruieren. Mit diesem Polynom berechnet der Betrüger den Wert $s_1^* = f^*(x_1)$ mit seiner Stützstelle x_1.
Sei nun $f_{\tilde{K}}(x) := f(x) + f^*(x)$, dann geht das Polynom $f_{\tilde{K}}$ durch den Punkt $(0, \tilde{K} = K \Leftrightarrow 1), (x_1, (s_1 + s_1^*))$ und wegen $f^*(x_2) = \cdots = f^*(x_k) = 0$ durch die echte Stützpunkte $(x_2, s_2), (x_3, s_3), \ldots, (x_k, s_k)$ der Wächter $W_2, W_3, \ldots, W_k$. Wenn der Betrüger den Punkt $(x_1, (s_1 + s_1^*))$ als Share ausgibt, so erhalten die ehrliche Wächter als Ergebnis der Rekonstruktion $\tilde{K} = K \Leftrightarrow 1$. Das ist ein legaler, allerdings falscher Schlüssel, wenn nicht $K = 0$ ist $(\Rightarrow \tilde{K} = \Leftrightarrow 1 \notin \mathbb{Z}_p)$.
Da In diesem Schema die Stützstellen x_j nicht öffentlich sind, kann der Betrüger das

Polynom $f^*(x)$ nicht interpolieren und damit den Punkt $(x_1, (s_1 + s_1^*))$ auch nicht berechnen. Daher kann der Betrüger die Stützstellen der Wächter $W_2, W_3, \ldots, W_k$ nicht anders als raten.

Die Sicherheit des Schemas

Nun untersuchen wir den Fall, daß $(k \Leftrightarrow 1)$ unehrliche Wächter sich gegen einen ehrlichen Wächter beispielsweise W_1 verschwören.
Die $(k \Leftrightarrow 1)$ Betrüger geben bei der Rekonstruktion ihre falschen Shares $S_2^* = (x_2, = s_2^*), S_3^* = (x_3, s_3^*), \ldots, S_k^* = (x_k, s_k^*)$ bekannt. Dabei benutzen sie auch ihre echte Stützstellen $x_2, x_3, \ldots, x_k$, damit nicht einer von ihnen gerade die Stüzstelle x_1 des ehrlichen Wächter W_1 in seinem gefälschten Share verwendet, womit der W_1 den Betrug gleich bei der Freigabe der $(k \Leftrightarrow 1)$ Shares entdecken könnte.
Auf diese Weise erfahren die $(k \Leftrightarrow 1)$ Betrüger erfahren das Share $S_1 = (x_1, s_1)$ von W_1, während der Wächter W_1 den falschen Schlüssel rekonstruiert. Nach Interpolationstheorem (Lit. angeben) gibt es für jeden Punkt $(\tilde{K}, 0)$ genau ein Polynom $f_{\tilde{K}}(x)$ höchstens vom Grad $(k \Leftrightarrow 1)$, das durch den Punkt $(\tilde{K}, 0)$ und die gefälschte Stützpunkte $S_2^* = (x_2, = s_2^*), S_3^* = (x_3, s_3^*), \ldots, S_k^* = (x_k, s_k^*)$ geht. Demzufolge berechnet Der Wächter W_1 mit den gefälschten Shares genau dann das Polynom $f_{\tilde{K}}(x)$ und damit den falschen Schlüssel, wenn das Polynom $f_{\tilde{K}}(x)$ das $f(x)$ an der Stelle x_1 schneidet. Für jeden weiteren Schnittpunkt muß für dessen Abzisse $f_{\tilde{K}}(z) = f(z)$ gelten.
Wenn wir $f^- := f_{\tilde{K}} = f$ setzen, so ist $f^-(z) = 0$ für höchstens $(k \Leftrightarrow 1)$ Stellen, da f^- höchstens vom Grad $(k \Leftrightarrow 1)$ ist. Also die beiden Polynom $f_{\tilde{K}}(x)$ und $f()$ schneiden sich höchstens in $(k \Leftrightarrow 1)$ Stellen.
Die Stelle x_1, bei der sich die Polynome schneiden müssen, damit der Wächter W_1 den Wert $\tilde{K}$ als Schlüssel rekonstruiert, muß eine der $(p \Leftrightarrow k)$ Zahlen $\{1, \ldots, p \Leftrightarrow 1\} \Leftrightarrow \{s_2^*, \ldots, s_k^*\}$ sein. Sei $\tilde{K}$ fest, so ist die Wahrscheinlichkeit, daß der Wächter W_1 diesen Wert als falscher Schlüssel berechnet, höchstens $\frac{k-1}{p-k}$.
Es gibt genau $(s \Leftrightarrow 1)$ legale, allerdings falsche Schlüssels, so daß die Wahrscheinlichkeit für einen eerfolgreichen Betrug höchstens $\frac{(s-1)(k-1)}{p-k}$ beträgt, d.h. es eysitiert eine Schranke $\varepsilon mit \varepsilon > \frac{(s-1)(k-1)}{p-k}$. Der Dealer kann daher eine sog. *Sicherheitsschranke* ε bestimmen und durch eine geeignete Wahl von der Werte p und s die Wahrscheinlichkeit für einen erfolgreichen Betrug noch kleiner als diese Schranke ε bestimmen.

Auch wenn ein Betrug in diesem Schema mit hoher Wahrscheinlichkeit entdeckt wird, kommt der Betrüger an die $(k \Leftrightarrow 1)$ Shares der ehrlichen Wächter. Um dies zu verhindern, „muß ein Betrug entdeckt werden, bevor der Schlüssel rekonstruiert wird“. Dazu veröffentlicht der Dealer zuerst eine Zahl $Ddummy$, bei der es sich um einen illegalen Schlüssel, beispielsweise $S_{dummy} = p \Leftrightarrow 2 \in [s, \ldots, p \Leftrightarrow 1]$, handelt. Danach sendet der Dealer jedem Wächter mehrere Shares $S_{i,j}$ $(i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m)$, mit denen sich die m Zahlen S_j rekonstruieren lassen. Sowohl die Shares $S_{i,j}$ als auch die Zahlen S_j werden durch den Index j in einer festen Reihenfolge festgesetzt. Für einen vom Dealer zufällig und geheim gewählten Index t handelt es sich bei S_t um den eigentlichen Schlüssel $S_t = K$. Die verbleibenden Zahlen S_j $(j \neq t)$ sind *Dummy-Zahlen* $S_j = S_{dummy}$.

Die Rekonstruktion führen die Wächter bezüglich der durch den Index j festgelegten Reihenfolge in diskreten Runden durch. Mit anderen Worten heißt es, die Wächter einer autorisierten Teilmenge berechnen in der $r \Leftrightarrow ten$ Runde mit ihren Shares $S_{i,r}$ den Wert S_r und können diesen überprüfen, bevor sie Ihre Shares $S_{i,r+1}$ offenbaren. Da es einem Betrüger nicht bekannt ist, in welcher Runde den eigentlichen Schlüssel rekonstruiert werden soll, muß er in jeder Runde betrügen, wenn er verhinder will, daß der Schlüssel von den ehrlichen Wächtern rekonstruiert werden, bevor er mit seinem Betrugsversuch begonnen hat. Da ein Betrugsversuch von ehrlichen Wächtern mit sehr hoher Waheschеinlichkeit, das ist $(1 \Leftrightarrow \varepsilon)$, entdeckt und in diesem Fall die Rekonstruktion abgebrochen werden können, ist es für einen Betrüger *praktisch* nicht möglich, mit gefälschten Shares in $(t \Leftrightarrow 1)$ Runden vor der Rekonstruktinsrunde (Runde t) unentdeckt bei seinem Betrugsversuch zu bleiben.
In [32] wird die Wahrscheinlichkeit angegeben, daß ein Betrüger in der Runde t an den Schlüssel dran zu kommen, ohne in den $(t \Leftrightarrow 1)$-vorherigen Runden unentdeckt zu bleiben, ist kleiner als $\dfrac{1}{t(1 \Leftrightarrow \varepsilon)}$.

Obwohl ein erfolgreicher Betrug auf diese Weise verhindert werden, können $(k \Leftrightarrow 1)$ ehrlich Wächter nicht in der Lage sein, den Cheater in diesem Verfahren zu identifizieren.
Um dieses Problem (theoretisch) zu lösen, sollen $(k+1)$ Wächter an die Rekonstruktion teilnehmen. Mit diesen $(k+1)$ Shares lassen sich $(k+1)$ Mengen von k Shares bilden, in denen das Share von jeweils einem Wächter nicht vertreten ist. Wenn es beiden $(k+1)$ Wächtern nur einen Betrüger gibt und die verbleibenden Wächter ehrlich sind, gibt es eine Menge von k Shares, mit denen sich ein legaler Schlüssel oder der Wert S_{dummy} ergibt.
Damit werden die k Shares dieser Menge als korrekte Shares bestätigt. Wenn jetzt $(k \Leftrightarrow 1)$ korrekte Shares zusammen mit einem anderen Share einen illegalen Schlüssel rekonstruieren, muß dieses Share gefälscht und der entsprechende Wächter betrüger sein. Auf diese Weise können r Betrüger identifiziert werden, wenn $(k+r)$ Wächter an die Rekonstruktion teilnehmen und k von diesen ehrlich sind.

Implementierungsbeispiel

Im folgenden Beispiel wird das erste *Recover-Protokoll* benutzt, wobei die Berechnung des Schlüssels auf die Lösung des linearen Gleichungssytems basiert, um zu zeigen, wie Betrugsversuch in diesem Schema (nicht) entdeckt werden kann.

Share-Verteilung.

Der Schlüssel $K = 3$ soll unter der Wächtermenge $\mathcal{W} = \{W_1, W_2, W_3, W_4, W_5\}$ geteilt werden. Aus Platzgründen sei

$$\mathcal{A}^0 = \{B = \{W_a, W_b, W_c\} \in \mathcal{A} : \ |B| = 3, \ a, b, c = 1, 2, \ldots, 5\}$$

die Basis von der Threshold-Zugriffsstruktur $\mathcal{A}$. In $\mathcal{A}^0$ gibt es exakt $\binom{5}{3} = 10$ autorisierte Teilmengen. Die autorisierte Teilmenge $\{W_1, W_2, W_3\} \in \mathcal{A}^0$ soll mit dem $(3,5)$-Threshold-Schema nach Tompa und Woll den Schlüssel $K = 3$ rekonstruieren. In $\{W_1, W_2, W_3\}$ wird Ehrlichkeit von W_1, W_2, W_3 nicht vorausgesetzt.

Dazu wählt eine Primzal $p = 23$. Alle Operationen sind im Restklasse Körper $\mathbb{Z}_{23}$. Weiter wählt der Dealer eine Zahl $s = 5 \ll p$, so daß in $[0, \ldots, 5]$ legale Schlüssels und in $[6, \ldots, 22]$ liegen. Demnach ist die Wahrscheinlichkeit für einen erfolgreichen Betrug höchstens $\varepsilon = 0.5 > \frac{10}{21}$.

Um die Shares für die Wächter zu bestimmen, wählt Dealer zufällig ein Polynom vom Grad 3 mit $f(x) = 3 + 4x + 5x^2 \in \mathbb{Z}_{23}[x]$ und 5 Stützstellen x_1, x_2, x_3, x_4, x_5, so daß sich als Shares:
$S_1 = (2, 8), \quad S_2 = (8, 10), \quad S_3 = (11, 8), \quad S_4 = (13, 3), \quad S_5 = (3, 14)$ ergibt.
Diese Shares schickt Dealer jeweils an die Wächter W_1, W_2, W_3, W_4, W_5. Damit ist die *Share-Verteilung* abgeschlossen.

Schlüssel-Rekonstruktion

Die Wächter W_1, W_2, W_3 bestimmen mit ihren Shares das Gleichungssystem:

$$\begin{aligned} 1c_0 + 2c_1 + 4c_2 &= 8 \quad G_1 \\ 1c_0 + 8c_1 + 18c_2 &= 8 \quad G_2 \\ 1c_0 + 11c_1 + 6c_2 &= 8 \quad G_3. \end{aligned}$$

Die Wächter lösen das Gleichungssystem mit dem Algorithmus nach *Gauß* und erhalten die Koeffizienten: $K = c_0 = 3, \quad c_1 = 4 \quad \text{und} c_2 = 5.$

Fall 1.
Im folgenden wird ein Betrugsversuch beschrieben, bei dem die Wächter W_2, W_3 gegen den ehrlichen Wächter W_1 verschwören, der ihren Betrug entdecken kann.
Da bei einer zufälligen Wahl der Stützstellen für die gefäschltex Shares die Möglicjkeit besteht, daß einer der Betrüger die Stützstelle von W_1, das ist $x_1 = 2$, wählt, so daß ihr Betrug gleich beim Aufdecken der gefälschte Shares, darunter in diesem Fall auch $x_1 = 2$, entdeckt werden kann (von W_1), verwenden sie ihre echten Stützstellen $x_2 = 8,\ x_3 = 11$. Da sie weiterhin die Stützstelle x_1 von W_1 nicht kennen, bleibt ihnen zur Bestimmung der Stützwerte nicht anders sie zu raten bzw. zufällig zu wählen. Die beiden Betrüger W_2, W_3 wählen zufällig als Stützwerte $s_2^* = 3\ s_3^* = 10$ und geben in der Rekonstruktionsphase die (gefälschten) Shares $S_2 = (8, 3)\ S_3 = (11, 10)$ frei. Zusammen mit seinem Share $S_1 = (2, 8)$ hat der ehrliche Wächter W_1 fälsches Gleichungssystem aufgestellt:

$$\begin{aligned} 1c_0 + 2c_1 + 4c_2 &= 8 \quad G_1 \\ 1c_0 + 8c_1 + 18c_2 &= 3 \quad G_2 \\ 1c_0 + 11c_1 + 6c_2 &= 10 \quad G_3. \end{aligned}$$

Die Lösung dieses Gleichungsystems *modulo p* führt zu: $\tilde{K} = c_0 = 17,\ c_1 = 8,\ c_2 = 11$. Mit dem Wert $\tilde{K} = 17 \in [6, \ldots, 22]$ weiß der ehrliche Wächter W_1, daß ein Betrug stattgefunden hat. Als Folge dieses Betruges hat W_1 den falschen Schlüssel rekonstruiert, während die Betrüger W_3, W_2 das Share $s_1 = (2, 8)$ von W_1 jetzt kennen und sie können damit den Schlüssel ohne W_1 rekonstruieren.

Fall 2.

Die Betrüger W_3, W_2 können auch die falschen Shares ausgeben, ohne daß sie ihre echten Stützstellen x_2, x_3 benützen. Die unehrlichen Wächter W_2, W_3 wählen ihre Shares zufällig, so daß sich bei der Rekonstruktion ein falscher, aber legaler Schlüssel ergibt. Der Dealer kann diese Gefahr zwar mit der Wahl von p und s beliebig klein machen, aber er kann sie nicht ausschließen. Die von W_2, W_3 zufällig gewählten Shares seien $S_2 = (8, 12),\ S_3 = (11, 21)$. Der ehrliche Wächter W_1 stellt anhand dieser beiden Shares zusammen mit seinem das Geichungssystem auf:

$$\begin{aligned} 1c_0 + 2c_1 + 4c_2 &= 8 \quad G_1 \\ 1c_0 + 8c_1 + 18c_2 &= 12 \quad G_2 \\ 1c_0 + 11c_1 + 6c_2 &= 21 \quad G_3. \end{aligned}$$

Als eindeutige Lösung ist: $\tilde{K} = c_0 = 4, \quad c_1 = 10$ und $c_3 = 19$. Da $\tilde{K} = 4 \in [0, \ldots, 5]$ ein legaler Wert ist, kann der Betrug in diesem Fall nicht entdeckt werden.

4.4.2 Das Verfahren von Lin und Harn

Es handelt sich um ein Secret Schema, welches allgemeine Zugriffsstruktur realisiert. Das Schema ist effizient, da jeder Wächter ein Share für die Rekonstruktion des Schlüssels. Das schema hat folgende Eigentschaften:

- Allgemeine Zugriffsstruktur und jeder Wächter bekommt einen Shares erhält.
- Betrugversuch entdecken und der Betrüger identifizieren.
- Auf unbewiesene kryptographische Annahme (*RSA*).
- mit nicht-autorisierte Teilmengen arbeiten (neu).
- mit dem Protokoll von Tompa und Woll [32] kombinieren, um zu verhindern, daß ein Betrüger bei seinem Betrug den Schlüssel aufdeckt.

Es sei $\mathcal{A}_\mathcal{P}$ positive und $\mathcal{A}_\mathcal{N}$ negative Zugriffsstruktur. Sei $n = p_1 \cdot p_2$, p_1 und p_2 seien zwei große Primzahlen und geheim. Der Schlüssel sei K und der Share von W_i sei S_i.Dann läßt das Schema wie folgt beschrieben:
Der Dealer bestimmt zunächst anhand $\mathcal{A}_\mathcal{N}$ einen Wert t_i deart, daß sich der größte gemeinsame Teiler (ggT) für die Werte der Wächter W_j einer jeden autorisierten Teilmenge als $ggT(t_{j1}, t_{j2}, \ldots, t_{jk}) = 1$ ergibt und andererseits für die Werte t_h der Wächter W_h von jeder unautorisierten Teilmenge aus $\mathcal{A}_\mathcal{N}$ als $ggT(t_{j1}, t_{j2}, \ldots, t_{jk}) = t > 1$ gilt.
Zu jedem Wächter wird dieser Wert t_i veröffentlicht. Jeder Wächter W_i erhält als Share zu dem Schlüssel K den Wert $S_i = K^{t_i} \mod n$. Mit diesen Shares können die Wächter einer autorisierten Teilmenge aus $\mathcal{A}_\mathcal{P}$ den Schlüssel K rekonstruieren, indem sie mit dem *erweiterten Algorithmus* zu den veröffentlichten Werten t_i die Zahlen a_i mit $\sum_{l=1}^{k} a_{jl} t_{jl} = 1$ und dann mit $\prod_{\nu=1}^{k} S_\nu^{a_{j\nu}} \mod n = K^{\sum_{l=1}^{k} a_{jl} t_{jl}} \mod n = K$

den Schlüssel K bestimmen. Die Wächter W_h einer unautorisierten Teilmenge aus $\mathcal{A}_\mathcal{N}$ können auf diese Weise nur $K^t \mod n$ berechnen.
Die Sicherheit des Schemas beruht darauf, daß K aus $S_i = K^{t_i} \mod n$ oder $K^t \mod n$ nicht in vernünftiger Zeit abgeleitet werden kann.

Mit diesem Schema beschreibt L. Harn und H.-Y. Lin, wie die allgemeine Zugiffstruktur durch das computational-sichere Secret Sharing Schema realisiert wird, wobei jeder Wächter ein Share erhält, Betrug entdeckt sowie Betrüger identifiziert werden kann. Das Schema kann mit Hilfe des Tompa und Woll [32] so erwetert werden kann, so daß der Betrüger den Schlüssel nicht erfährt. Dies wird durch die folgenden Protokolle beschrieben:

Das Protokoll *share*

1. Dealer bestimmt die negative Zugriffsstruktur $\mathcal{A}_\mathcal{N}$ als die Menge aller Wächtergruppe, die nicht dazu berechtigt sind, den Schlüssel K zu rekonstruieren. Diese Wächtergruppen werden als N_i. Dann wälht er zufällig eine Zahl $n > K$ mit $n = p_1 \cdot p_2$. Es sind zwei geheime große Primzahlen. Die Rechnung in diesem Schema sind *modulo n*.

2. Es sei $\mathcal{M}(\mathcal{A}_\mathcal{N}) = \{N_j \in \mathcal{A}_\mathcal{N} \mid \forall N_h \in \mathcal{A}_\mathcal{N} \quad \text{und} \quad N_j \nsubseteq N_h\}$ maximierte negative Zugriffsstruktur. Der Dealer ordnet jedem $N_j \in \mathcal{M}(\mathcal{A}_\mathcal{N}$ eine andere Primzahl q_j zu.

3. Dealer berechnet für jeden Wächter W_i eine Wert t_j als Produkt der Primzahlen q_j aller nicht-autorisierten Teilmengen N_i aus $\mathcal{M}(\mathcal{A}_\mathcal{N})$, in denen der Wächetr W_i partuzipiert ist. Formal gilt: $t_i = \prod_{W_i \in N_j} q_j$.

4. Dealer berechnet für jeden Wächter W_i als Share $S_i = K^{t_i} \mod n$ und überprüft, ob $p_1 | S_i$ oder $p_2 | S_i$ gilt. Falls es ein der beiden Fälle zutrifft, dann muß er die Schritte 1.,2. und 3. mit neuen Primzahlen q_j wiederholen.

5. Dealer verteilt die S_i auf die Wächter W_i und veröffentlich die Weter t_i und dies Zahl n.

Mit dem folgenden Protokoll können k Wächter $W_{i,1}, \ldots, W_{i,k}$ der autorisierten Teilmenge $N_i \in \mathcal{A}_\mathcal{P}$ den Schlüssel rekonstruieren.

Das Protokoll *recover*

1. Die Wächter $W_{i,j}$ der autorisierten Teilmenge N_i geben ihre Shares $S_{i,j}$ den anderen an die Rekonstruktion beteiligten Wächtern bekannt (Siehe Seite 39). Danach kann jeder Wächter überprüfen, ob sich unter diesen Shares ein gefälschtes Share befindet.

2. Jeder Wächter $W_{i,j}$ berechnet mit dem erweiterten Euklidischen Algorithmus die Zahlen $a_1, \ldots, a_k$ mit $a_1 t_{i,1} + \cdots + a_k t_{i,k} = 1 = \text{ggT}(t_{i,1}, \ldots, t_{i,k})$.

3. Jeder der Wächter $W_{i,j}$ bestimmt den Schlüssel K mit
   $S_1^{a_1} \cdot \ldots \cdot S_k^{a_k} \mod n = K^{a_1 t_1 + \cdots a_k t_k} \mod n = K$.

Die Funktionweise des Schemas basiert darauf, daß für die Werte t_i von jeder autorisierten Teilmenge $\mathrm{ggT}(t_{i,1}, \ldots, t_{i,k}) = 1$ und für die von jeder unautorisierten Teilmenge $\mathrm{ggT}(t_{i,1}, \ldots, t_{i,k}) > 1$ gilt. Um diese Bedigungen zu erfüllen, werden für jeder autorisierte Teilmenge eine andere Primzahl q_j gewählt und der Wert t_i für den Wächter W_i als Produkt derjenigen Primzahlen berechnet, deren autorisierten Teilmengen den Wächter W_i enthalten. Aus diesem Ansatz läßt sich mit der folgenden Überlegung eine effiziente Methode zur Berechnung von t_i herleiten.
Wenn die Shares einer Gruppe von Wächtern zur Rekonstruktion des Schlüssels K nicht ausreichen, dann kann eine Teilgruppe dieser Wächter mit ihren Shares den Schlüssel nicht bestimmen, so daß jede Teilmenge einer nicht-autorisierten Teilmenge ebenfalls einer nicht-autorisierten Teilmenge darstellen muß. Dieser Gedanke liegt in der Konstruktion von der maximierten negativen Zugriffsstruktur $\mathcal{M}(\mathcal{A}_\mathcal{N}) = \{N_j \in \mathcal{A}_\mathcal{N} \mid \forall N_h \in \mathcal{A}_\mathcal{N}\ N_j \nsubseteq N_h\}$ zugrunde. Diese Menge hat weniger Elemente als die negative Zugriffsstruktur $\mathcal{A}_\mathcal{N}$. Wenn der Dealer $\mathcal{M}(\mathcal{A}_\mathcal{N})$ verwendet, um die t_i zu berechnen, muß er weniger Primzahlen bestimmen, so daß sich kleinere Werte t_i ergeben.
Insgesamt können die Berechnung des Schemas einfacher und schneller durchgeführt werden.

Um die Korrektheit des Schemas zu zeigen, betrachten wir für eine Gruppe von Wächtern $\{W_1, \ldots, W_k\}$ die entsprechende Werte $t_1, \ldots, t_k$ mit $t = \mathrm{ggT}(t_1, \ldots, t_k)$. Für diese Werte können die Wächter mit dem erweiternten Euklidischen Algorithmus die Zahlen $a_1, \ldots, a_k$ mit $a_1 t_1 + \cdots + a_k t_k = t$ berechnen. Zusammen mit ihren Shares $S_i = K^{t_i} \mod n$ köennen sie die folgende Gleichung aufstellen:

$$S_1^{a_1} \cdot \ldots \cdot S_k^{a_k} \mod n = K^{a_1 t_1 + \cdots + a_k t_k} \mod n = K^t \mod n. \tag{4.15}$$

Mit dieser Gleichnug kann der Schlüssel von den Wächtern $W_1, \ldots, W_k$ rekonstruiert werden, wenn die Werte $t_1, \ldots, t_k$ teilerfremd sind, also $t = \mathrm{ggT}(t_1, \ldots, t_k) = 1$ gilt.

Im folgenden seien $W_1, \ldots, W_k$ die Wächter einer autorisierten Teilmenge. Wir nehmen an, daß sich der Schlüssel K nicht rekonstruieren läßt, so daß $\mathrm{ggT}(t_1, \ldots, t_k) = t > 1$ gelten müsse.
Für jeden Wächter W_i wird der Wert t_i als Produkt derjenigen Primzahlen q_j berechnet, die den nicht-autorisierten Teilmengen zugeordnet sind, in denen jeweils der Wächter W_i partizipiert ist. Damit läßt sich $\mathrm{ggT}(t_1, \ldots, t_k)$ als Produkt von den q_j, die als Faktoren in jedem der Werte $t_1, \ldots, t_k$ vertreten sind. Wenn eine Primzahl q_j als Faktor in allen Werten der Wächter $W_1, \ldots, W_k$ vorkommen, ist jeder dieser Wächter in dem Element der Menge $\mathcal{M}(\mathcal{A}_\mathcal{N})$ enthalten, dem die Primzahl q_j zugeordnet ist. Die Wächtergruppe $W_1, \ldots, W_k$ ist daher eine Teilmenge des Elements der Menge $\mathcal{M}(\mathcal{A})_\mathcal{N}$ und stellt eine nicht-autorisierte Teilmenge dar. Da eine Wächtergruppe nicht gleichzeitig als autorisierte und nicht-autrisierte Teilmenge darstellt, ist dies einen Widerspruch zu der Annahme. Damit ist die Korrektheit des Schema bewiesen.

Für die praktische Berechnung des Schlüssels K wird wie folgt erläuchtert: Zu den Werten $t_1, \ldots, t_k$ werden die Zahle $a_1, \ldots, a_k$ mit $\sum_{\nu=1}^{k} a_\nu t_\nu = 1 = \text{ggT}(t_1, \ldots, t_k)$ durch die Wächter einer autorisierten Teilmenge bestimmt.
Mit der Gleichnung $S_1^{a_1} \cdot \cdots \cdot S_k^{a_k} \mod n = K$ können die Wächter den Schlüssel K rekonstruiern. Dabei können sich einige der Zahlen $a_1, \ldots, a_k$ allerdings negative Werte ergeben, so daß Faktoren mit negativen Exponenten $(S_i^{-a_i})$ in der Gleichung auftreten. Um dies zu bedandeln, wird $S_i^{a_i} = (S_i^{-1})^{a_i}$ umgeformt. Dabei S_i^{-1} ist das inverse Elemnt S_i *modulo* n. Da Exsistenz eines inversen Elements nur für Zahlen garantiert werden kann (Kapitel 2), die zu n teilerfremd sind, muß im Schritt 4. des Schemas für jedes Share S_i die Gleichung $\text{ggT}(S_i, n) = 1$ gelten. Für das Inverse Element S_i^{-1} zu S_i muß gelten $S_i^{-1} S_i \mod n = 1$. Wegen $S_i^{-1} S_i \mod n = 1 \Leftrightarrow S_i^{-1} S_i = \alpha n + 1 \Leftrightarrow S_i^{-1} S_i + (\Leftrightarrow \alpha) n = 1$ läßt sich der erweiterte Euklidische Algorithmus anwenden, um das Inverse Element S_i^{-1} *modulo* n zu berechnen. Der Aufruf $\text{ggT}_{ext}(n, S_i)$ liefert als Ergebnis $(1, a_1, a_2)$ mit $a_1 n + a_2 S_i = 1 \Leftrightarrow a_2 S_i \mod n = 1$. Demnach gilt

$$S_i^{-1} = \begin{cases} a_2 & \text{für positives} \quad a_2 \\ a_2 + n & \text{für negatives} \quad a_2 \end{cases}$$

Die Sicherheit des Schemas
Das Schema ist computational-sicher. Ihre Sicherheit beruht auf die RSA-Annahme. Demnach kann der Schlüssel K nicht vernünftiger Zeit berechnet werden, wenn die Faktorisierung von n, also die beiden Primzahlen nicht bekannt sind.
Es sei $\{W_1, \ldots, W_k\} \in \mathcal{A}_\mathcal{N}$ eine nicht-autorisierte Teilmenge. Aus der Definition von $\mathcal{M}(\mathcal{A}_\mathcal{N}) = \{N_j \in \mathcal{A}_\mathcal{N} \mid \forall N_h \in \mathcal{A}_\mathcal{N} \quad \text{und} \quad N_j \not\subseteq N_h\}$ muß diese nicht-autorisierte Teilmenge eine Teilmenge von mindestens einem Element von $\mathcal{M}(\mathcal{A}_\mathcal{N})$ sein. Damit in jedem der Werte $t_1, \ldots, t_k$ die Primzahl q_j als Faktoren vertreten, welche diesem Element zugeordnet wird. Da die Werte $t_1, \ldots, t_k$ nicht teilerfremd sind, muß $\text{ggT}(t_1, \ldots, t_k) = t > 1$ sein. Die Wächter $W_1, \ldots, W_k$ können dann mit $S_1^{a_1} \cdot \cdots \cdot S_k^{a_k} \mod n$ höchstens eine Wert $S^t \mod n$ berechnen. Nach RSA-Annahme kann der Schlüssel K weder von $S^t \mod n$ noch von $S_i^{t_i} \mod n$ abgeleitet werde.

Betrug-Erkennung, Betrüger-Indentifikation

Warum ist diese Behandlung wichtig?. In traditionellen Secret Sharing Schemes kann ein Wächter W_i die anderen betrügen, so daß er anstelle seines korrekten Shares ein falsches Share freigibt. Die ehrlichen Wächter rekonstruieren den falschen Schlüssel, während der Betrüger den korrekten Schlüssel K mit seinem korrekten Share zusammen mit anderen Shares, die erfahren hat.
IN diesem Schema wird nicht nur Betrugsversuch entdeckt, sondern auch die werden Betrüger identifiziert. Mit Eine Kommbination mit dem Schema von Tompa und Woll wird zusätzlich verindert, daß der cheater den Schlüssel beim Betrugsversuch nicht erfahren kann.

Nun gehen wir davon aus, daß die beiden Wächter W_1 und W_2 ihre Shares für die Rekonstruktion des Schlüssels K ausgeben sollen. Dann kann der Wächter W_1 über-

prüfen, ob sich bei dem vom Wächter W_2 ausgegebenen Wert S_2^* um das richtige Share S_2 oder ein falsches Share handelt. Der Wächter W_i berechnet mit dem erweiterten Algorithmus zuerst den grösten gemeinsamen Teiler t für die öffentliche Werte t_1, t_2 und dann die Zahlen a_1, a_2 mit $a_1 t_1 + a_2 t_2 = ggT(t_1, t_2)$. Wenn der Wächter W_2 seinen echten Share S_2 ausgibt, dann gilt: $S_2^* = S_2 = S^{t_2} \mod n$. Dann kann der Wächter W_1 den Wert K^t mit der Gleichung

$$S_1^{a_1} \cdot \cdots \cdot (S_2^*)^{a_2} \mod n = K^{a_1 t_1 + a_2 t_2} \mod n = K^t$$

berechnen. Wegen $(K^t)^{\frac{t_1}{t}} \mod n = K^{t_1 \mod n} = S_1$ kann der Wächter W_1 das Ergebnis der ersten Gleichung anhand seines Share S_1 überprüfen.
Wenn der Wächter W_2 versucht, mit dem gefälschten Share $S_2^* \neq S_2$ zu betrügen, erhält der Wächter W_1 mit $S_1^{a_1} S_2^{*a_2} \mod n = K^* \neq K^t$ einen Wert als Ergebnis, das sich bei der Prüfung $(K^*)^{\frac{t_1}{t}} \mod n$ als falsch ergibt. Damit kann ein ehrlicher Wächter jeden Betrugsversuch mit Sicherheit entdecken und den Betrüger identifizieren.

Wie erwähnt, es muß sicher gestellt werden, daß der Betrüger den Schlüssel beim Betrugsversuch nicht in Erfahrung bringen kann. Dazu wird eine Variante des Schemas von Tompa und Woll verwendet. Der Dealer sendet jedem Wächter W_i die m Shares $S_{i,j}$, mit denen die Zahlen S_j, $1 \leq i \leq n$, $1 \leq j \leq m$. Die S_j werden vom Dealer so gewählt, daß für den Index t eine der Zahlen S_t der Schlüssel $S_t = K$ ist und $S_{j-1} < S_j$ für $j < t$ und $S_{t-1} > S_t$ gilt. So werden die Rekonstruktion in t geheimen Runden vorgenommen, in denen jeweils eine der Zahlen S_j bezüglich ihrer mit dem Index j bestimmten Reihenfolge berechnet wird. Wenn die Wächter eine Zahl S_j bestimmen, welche kleiner als S_{j-1} der vorherigen Runde ist, haben sie damit den Schlüssel $K = S_j < S_{j-1}$ rekonstruiert. Da ein Betrüger nicht weiß, in welcher Runde der (eigentlichen) Schlüssel rekonstruiert werden soll, muß er für seinen Betrug eine Runde auswählen. Die a priori Wahrscheinlichkeit für einen Betrug in einer kritischen Runde, mit dem der Betrüger, jedoch nicht die ehrlichen Wächter, den Schlüssel in Erfahrung zu bringen, beträgt $\frac{1}{m}$. Auch in diesem Fall kann der Betr!uger von ehrlichen Wächtern mit identifiziert werden.

Zahlenbeispiel

Es sei $W = \{W_1, W_2, W_3, W_4\}$ die Wächtermenge. Der Schlüssel sei $K = 3$. Der Schlüssel soll auf die vier Wächter verteilt werden. Der Schlüssel K soll durch eine Wächtergruppe, welche nicht zur negativen Zugrffsstruktur angehört, rekonstruiert werden.

Share-Verteilung.

Der Dealer initialisiert die Verteilungsphase mit folgenden Schritten:

1. Die negative Zugriffsstruktur sei
   $\mathcal{A}_{\mathcal{N}} = \{ \{W_1\}, \{W_2\}, \{W_3\}, \{W_4\}, \{W_2, W_3\}, \{W_2, W_4\} \{W_3, W_4\} \}$. Dealer wählt zwei Primzahl $p_1 = 7$ und $p_2 = 11 \Rightarrow n = 7 \cdot 11 = 77$ als Basis der Restklassekörper *modulo* 77, bezeichnet als $\mathbb{Z}_{77}$.

2. Der Dealer bildet die maximierte negative Zugriffsstruktur als $\mathcal{M}(\mathcal{A}_\mathcal{N}) = \{\{W_1\}, \{W_2, W_3\}, \{W_2, W_4\}, \{W_3, W_4\}\}$. Den Elementen von $\mathcal{M}(\mathcal{A}_\mathcal{N})$ werden durch die Bijektion $\varphi : \mathcal{M}(\mathcal{A}_\mathcal{N}) \to \{2, 3, 5, 7\}$ zugeordnet, so daß $\varphi(W_1) = 2,\ \varphi(\{W_2, W_3\}) = 3,\ \varphi(\{W_2, W_4\}) = 5,\ \varphi(\{W_3, W_4\}) = 7$ festgelegt sind. Zu jedem Wächter W_i berechnet der Dealer die entsprechenden Werte t_i als Produkt der Primzahlen der Elemente von $\mathcal{M}(\mathcal{A}_\mathcal{N})$, in denen der Wächter W_i partizipiert ist. Das sind: $t_1 = 2 \quad t_2 = 3 \cdot 5 = 15 \quad t_3 = 3 \cdot 7 = 21 \quad t_4 = 5 \cdot 7 = 35$.

3. Der Dealer berechnet nun die Shares s_i für W_i mittels $S_i = K^{t_i} \mod 77$:
   $S_1 = 3^2 \mod 77 = 9 \qquad S_2 = 3^{15} \mod 77 = 34$
   $S_3 = 3^{21} \mod 77 = 69 \quad S_4 = 3^{35} \mod 77 = 12$
   Da eine der Bedingung $p_1|S_i$ oder $p_2|S_i$ nicht erfüllt ist, ist keine Wiederholung mit neuen Primzahlen q_j nicht notwendig.

4. Der Dealer sendet S_i als Shares an die Wächter W_i und veröffentlicht die Werte t_i und die Zahl n.

Schlüssel-Rekonstruktion.

Wir betrachten die Wächtergruppe $\mathcal{P} = \{W_2, W_3, W_4\}$. Da die Menge $\mathcal{P} \notin \mathcal{A}_\mathcal{N}$ ist, bildet sie eine autorisierte Teilmenge, in denen die Wächter W_2, W_3, W_4 berechtigt sind, den Schlüssel K zu rekonstruieren.

Die Wächter benutzen den erweiterten Euklidischen Algorithmus (EEA), um für die öffentliche Werte $t_4 = 25$ und $t_3 = 21$ den größten gemeinsamen Teiler t_h und die Zahlen $a_{1,1}$ und $a_{1,2}$ mit $a_{1,1}t_4 + a_{1,2}t_3 = t_h$ zu berechnen.

t_4	t_3	$\frac{t_4}{t_3}$	$a_{1,1}$	$a_{1,2}$	t_h
35	21	1	-1	2	7
21	14	1	1	-1	7
14	7	2	0	1	7
7	0	-	1	0	7

Anhand der obigen Tabelle liefert der EEA $t_h = 7$ und die Werte $a_{1,1} = \Leftrightarrow 1$ und $a_{1,2} = 2$ mit Teiler t_h und die Zahlen $a_{1,1}$ und $a_{1,2}$ mit $a_{1,1}t_4 + a_{1,2}t_3 = (\Leftrightarrow 1 \cdot 35 + 2 \cdot 21 = 7 = t_h)$.

Wie der Wächter das Share überprüft, das der Wächter W_4 ausgegeben hat, wird wie folgt verdeutlicht: Dazu berechnet Wächter W_1 den Ausdruck $K^{t_h} \mod 77$ (W_3 weißt nicht von K) mit Hilfe der Gleichung $S_4^{a_{1,1}} + S_3^{a_{1,2}} \mod n = 12^{-1} \cdot 69^2 \mod 77 = K^{t_h} \mod 77$.
Das zu 12 inverse Element *modulo* 77 wird mit dem EEA mit Hilfe der folgenden Tabelle berechnet.

n	S_3	$\frac{n}{S_3}$	a_1	a_2	t
77	12	6	5	-32	1
12	5	2	-2	5	1
5	2	2	1	-2	1
2	1	2	0	1	1
1	0	-	1	0	1

Als inverses Element zu 12 ergibt sich $12^{-1} \mod 77 = 77 \Leftrightarrow 32 = 45$. Wächter W_3 erhält nun $(K^{t_h}) \mod 77 = 45 \cdot 69^2 \mod 77 = 214245 \mod 77 = 31$ und überprüft diesen Wert mit $(K^{t_h})^{fract_3 t_h} \mod n = 31^3 \mod 77 = 69 = S_3$. Damit weißt W_3, daß W_4 das richtiges Share bekommen hat (bzw. ausgegeben hat).

Auf diese Weise kann natürlich auch W_4 das Share von W_4 auf seine Korrektheit überprüfen. Durch erneuten Anwendung des EEA berechnen die Wächter den größten gemeinsamen Teiler t von $t_h = 7$ und $t_2 = 15$ und zwei Zahlen $a_{2,1}$ und $a_{2,2}$ mit $a_{2,1}t_2 + a_{2,2}t_h = t$. Durch folgende Tabelle

t_2	t_h	$\frac{t_2}{t_h}$	$a_{2,1}$	$a_{2,2}$	t
15	7	2	1	-2	1
7	1	7	0	1	1
1	0	-	1	0	1

erhalten sie Wächter als Ergebnis $\text{ggT}(t_2, t_h) = 1$ und damit $\text{ggT}(t_2, t_3, t_4) = 1$ und die Werte $a_{2,1} = 1,\ a_{2,2} = \Leftrightarrow 2$ mit $a_{2,1}t_2 + a_{2,2}t_h = 1 \cdot 15 \Leftrightarrow 2 \cdot 7 = 1 = t$.

Wie (auf Seite 35) beschrieben, berechnen sie die Zhlen a_1, a_2, a_3 als
$a_2 = a_{2,1} = 1, \quad a_3 = a_{2,2}a_{1,2} = \Leftrightarrow 2 \cdot 2 = \Leftrightarrow 4, \quad a_2 = a_{2,1} = 1 \quad \text{und} \quad a_4 = a_2 = a_{2,2}a_{1,1} = (\Leftrightarrow 2)(\Leftrightarrow 1) = 2$ mit $t = a_2t_2 + a_3t_3 + a_4t_4 = (1 \cdot 15) + (\Leftrightarrow 4 \cdot 21) + (2 \cdot 35) = 1$.

Mit den Zahlen a_2, a_3, a_4 und ihren Shares können die Wächter in $\mathcal{P} = \{W_2, W_3, W_4\}$ den Schlüssel K wie folgt rekonstruieren:

$$K = S_2^{a_2} \cdot S_3^{a_3} \cdot S_4^{a_4} \mod n = 34^1 \cdot (691 \Leftrightarrow 1)^4 \cdot 12^2 \mod 77.$$

Der EEA wird nochmal angewendet, um das zu 69 inverse Element $\mod 77$ zu berechnen. Aus der folgenden Tabbelle wird als Ergebnis $69^{-1} \mod 77 = 77 \Leftrightarrow 29 = 48$ berechnet.

n	S_3	$\frac{n}{S_3}$	a_1	a_2	t
77	69	1	26	-29	1
69	8	8	-3	26	1
8	5	1	2	-3	1
5	3	1	-1	2	1
3	2	1	1	-1	1
2	1	2	0	1	1
1	0	-	1	0	1

Das inverse Element zu 69 ist nun $69^{-1} \mod 77 = 77 \Leftrightarrow 29 = 48$. Schließlich wird der Schlüssel K als Ergebnis durch $\mathcal{P} = \{W_2, W_3, W_4\}$ rekonstruiert mit:
$K = 34 \cdot 48^4 \cdot 12^2 \mod 77 = 34 \cdot 36 \cdot 67 \mod 77 = 82008 \mod 77 = 3$. *

4.4.3 Verifiable Secret Sharing Schemes

Probleme bei praktischen Einsätzen von Secret Sharing Schemes sind *Korrektheit* und *Konsistenz* der Shares. Traditionele Secret Sharing Schemes setzen stets voraus, daß es ein sicheres Kommunikationskanal gibt, damit die Shares bei der Share-Verteilung vom Dealer sicher zu Wächtern gesendet werden können. Übertragungsfehler, die durch Verwendung eines physikalischen Kanals entstanden sind, sind nicht Gegenstand in diesem Abschnitt. Daher nehmen wir an, daß es einen fehlererkennenden bzw. -korrigierenden Code gibt [33], so daß die Verfälschung der Shares seitens des verwendeten Kommunikationskanals ausgeschlossen werden können. Durch Einsatz kryptographischer Primitiven (Public-Schlüssel-Verschlüsselung und digitale Signatur) können erreicht werden, daß die Integrität der Shares (Geheimhaltung und Authetizität) gewährleistet sind. Es liegt nun an die Charakteristika der beteiligten Instanzen in einem Secret Sharing Schema. Das sind der Dealer und die Wächter. Ein unehrlicher Dealer kann falsche Shares auf die Wächter verteilen und unehrliche Wächter können behaupten, daß sie unkorrekte Shares erhalten haben. Diese Situationen können nicht ausgeschlossen werden, auch wenn die Ehrlichkeit der Instanzen vorausgesetzt sind. Ein schwaches Check-Protokoll kann die Redundanz, charakterisiert durch die verfügbare Shares und damit die autorisierten Teilmengen, der Schlüsselrekonstrukton vermindern. Gibt ein ehrlicher Wächter aus einer autorisierten Teilmenge seine Shares frei, um zusammen mit verbleibenden Wächtern den Schlüssel zu rekonstruieren, so wird dieser Wächter als Betrüger betrachtet, denn sein ist Share unkorrekt. Kommt dies oft vor, so bedeutet z.B. im (k, n)-Threshold-Schema von Shamir Verlust an autorisierten Teilmengen. Andere Betrachtung ist wiederholte Shares-Verteilungen bzw. Neuverteilung des Shares in einigen Anwendungen nicht erwünscht, denn in jedem Secret Sharing Schema ist im Bezug auf der Verwendung von Zugriffsstrukturen und Rekonstruktionsmethoden speicher-, rechenintensiv und kommunikationsaufwendig.
Auch selbst ein Wächeter mittels eines Check-Protokolls seine eignen Shares überprüfen kann, ist das Konsistenzsproblem der Shares in einer autorisierten Teilmenge noch nicht gelöst. Wie kann ein Wächter sicher sein, daß die Shares der verbleibenden Wächter korrekt sind. In der Tat gibt es Anwendungen, in denen es unter den Wächter einen sog. *Leader* gibt und er ist für die Rekonstruktion des Schlüssels verantwortlich ist. Vor jeder Rekonstruktion muß der Leader sicher sein, daß die Gesamtheit der Shares in der autorisierten Teilmengen konsistent sein, ohne die Shares zu offenbaren. Es soll daher ein Schema geben, mit dem jeder Wächter dem Leader (In Pedersen Prover) bzw. der Leader selbst beweisen kann, so daß die Konsistenz der Shares gewährleistet werden.

Das verifiable Secret Sharing Schema nach Pedersen Wie kann sich ein Wächter überzeugen kann, daß er ein korrektes Share erhält und nachdem er geprüft hat, daß

er ein korrektes Share erhalten hat, wie kann er sicher sein, daß sein Share mit den anderen konsistent ist, d.h. wie kann ein Wächter sicher sein, daß die Shares der verbleibenden Wächter, mit denen er den Schlüssel rekonstruieren kann, auch korrekt sind ?
Diese Proplematik beschätigen sich (Literatur hier eintrahen)

Bei dem Schema von Pedersen [23] handelt es sich um ein (k, n)-Threshold Schema von Shamir [25] mit der zusätzlicher Eigenschaft, daß jeder Wächter in $\mathcal{W} = \{W_1, \ldots, W_n\}$ sein Share überprüfen kann, ob er ein korrektes erhalten hat und darüber hinaus ob sein Share mit denen der anderen Wächter konsistent ist.
Die Sicherheit des Schema ist computational, denn es basiert auf der kryptographische Annahme, daß die Berechnung des diskreten Logarithmus in einem Restklassekörper $\mathbb{Z}_p$, p ist eine Primzahl, ein schwieriges Problem darstellt, welches für eine große Primzahl p nicht in „vernünftiger“ Zeit gelöst werden kann.
Nun soll das Schema von Pedersen realisiert werden. Das Schema besteht aus drei Protokollen. Das hier beschriebene Protokoll *share* unterscheidet sich von dem im Schema von Shamir, daß u.a. die Nummer i der Wächter W_i wegen Betrugsgefahr durch die vom Dealer gewählten Stützstellen x_i ersetzt (falsch geschrieben, da in diesem Verfahren x_i öffentlich sind) werden. Die Rekonstruktion in diesem Schemas wird durch das Protokoll *recover* des Schemas von Shamir. Neu ist das Überprüfungsprotokoll *verify-share*, mit dem die Korrekheit und Konsistenz der verteilten Share geprüft werden können.

Das Protokoll *share*

1. K sei der zu rekonstruierende Schlüssel. Dealer veröffentlicht zwei große Primzahlen $p > K$ und $q > K$ mit $q|(p \Leftrightarrow 1)$. Die von p und q bestimmten Raesklassenkörper seien entsprechend $\mathbb{Z}_p$ und $\mathbb{Z}_q$. Weiter veröfentlicht er den Generator $g \in \mathbb{Z}_p$, der die Untergruppe G in $\mathbb{Z}_p$ der Ordnung q.

2. Der Dealer wählt zufällig ein polynom $f(x) \in \mathbb{Z}_q[x]$ vom Grad $(k \Leftrightarrow 1)$ mit $f(x) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1}$ und $K = c_0$.

3. Für jedes c_j $(j = 1, 2, \ldots, (k \Leftrightarrow 1))$ berechnet der Dealer den Wert $C_j = g^{c_j} \mod p$ im Körper $\mathbb{Z}_p$.

4. Dealer wählt zufällig n verschiedene $x_1, \ldots, x_n \in \mathbb{Z}_q$ und berechnet dann für jeden Wächter W_i $(i = 1, 2, \ldots, n)$ als **Share** den Wert $s_i = f(x_i)$ im Körper $\mathbb{Z}_q$ und den Wert $S_i = g^{s_i} \mod p$ im Körper $\mathbb{Z}_p$.

5. Dealer

   - sendet jedem Wächter W_i das Share s_i und
   - veröffentlicht
     - die Werte S_i und x_i für jeden Wächter W_i und
     - die Werte C_j für die Koeffizienten c_j des Polynoms $f(x)$.

Das Protokoll *verify-share*

1. Jeder der Wächter W_i überprüft
   (a) sein Share s_i mit den öffentlichen Informationen S_i, p und g, indem er g^{s_i} mod p berechnet und mit S_i vergleicht,
   (b) für jeden Wächter W_j $(j = 1, 2, \ldots, n)$ die öffentliche Information S_j, indem er berechnet
   $\prod_{h=0}^{k-1} C_h^{x_j^h} = g^{c_0 + c_1 x_j^1 + \cdots + c_{k-1} x_j^{k-1}}$ mod p und mit S_j vergeleicht.
2. Der Wächter W_i nimmt sein Share s_i an, wenn alle berechneten Werte mit den vom Dealer veröffentlichten Werten übereinstimmt. Sonst veröffentlicht er zusammen mit dem empfanenen Share s_i eine Beschwerde.

Wenn sich zu viele Wächter beschweren, wird das Schema abgebrochen. Und der Dealer muß untersucht werden.

Im folgenden werden wieder zwei Rekonstruktionsmethoden beschrieben. Die Erste Methode basiert auf der Lösung des linearen Gleichungssystems und die zweite auf der Lagrange'schen Interpolationsformel im $\mathbb{Z}_p$.

Das Protokoll *recover*

1. Jeder der an der Rekonstruktion beteiligten Wächter W_j $(j = 1, 2, \ldots, k)$ gibt sein Share s_j den anderen Wächtern bekannt. Danach stellt jeder dieser Wächter mit den k Shares $s_j = f(x_j)$ und mit den öffentlich bekannten Stützstellen x_j die Gleichungen $G_j: \quad s_j = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1}$ auf.
   Jeder der k Wächter bekommt ein lineares Gleichungssystem mit den k Gleichungen und k unbekannten Koeffizienten $c_0, c_1, \ldots, c_{k-1}$.
2. Jeder Wächter W_j kann mit dem Gauß-Algorithmus das Gleichungssystem lösen. Die eindeutige Lösung dieses Gleichungssystems liefert $K = c_0$.

Das Protokoll *recover* (Lagrange'sche Interpolationsformel)

1. Jeder der Wächter in einer autorisierten Teilmenge W_j $(j = 1, 2, \ldots, k)$ gibt sein Share s_j an den anderen $k \Leftrightarrow 1$ Wächtern bekannt.
2. Jeder Wächter extrapoliert den Werten als Eingabe für den Algorithmus zur Lagrange-Interpolation den Schlüssel K als Wert des Polynoms an der Stelle $x = 0$.

Wenn es keine Kommunikationsfehler bei der Übermittlung der Shares gibt, so verlaufen die Rekonstruktion des Schlüssels in diesem Schema wie im schema von Shamir. Die Korrektheit dieses Schema kann mit denselben Argumenten hergeleitet werden.

Allerdings muß hier gezeigt werden, wie ein Wächter in diesem Schema überprüfen kann, daß sei Share korrekt ist und daß sein Share mit den den der anderen Wächtern in der autorisierten Teilmenge konsistent ist. Formal wird die Konsistenz der Shares als *k-Konsistenz* bezeichnet. Die Shares sind *k-konsistent*, wenn k Shares der autorisierten Teilmenge in der vorgegebenen Zugriffsstruktur den Schlüssel K eindeutig rekonstruieren.
IN dem Schema von Pedersen muß der Dealer zu jedem Share s_i einen Wert S_i mit $S_i = g^{s_i} \mod p$ veröffentlichen. Jeder der Wächter W_i kann anhand dieser veröffentlichten Information kontrollieren, ob er das korrekte Share s_i bekommen hat. Er überprüft mit den öffentlichen Werten p und g und dem empfangenen Wert s_i^*, ob die Gleichung

$$g^{s_i^*} \mod p = S_i = g^{s_i} \mod p$$

gilt. Falls die Gleichung gilt, akzeptiert der Wächter das empfangene Share s_i^* als korrektes Share s_i. Sonst veröffentlicht er die Beschwerde, daß er das falsche Share bekommen hat. Bei der Implementierung muß fest gelegt werden, wie der Dealer und die Wächter mit solchen Beschwerden umgehen können. Wenn sich zu viele Wächter beschweren, wird das Protokoll angebrochen und der Dealer ist unehrlich.
Wenn sich anderseits ein Wächter wiederholt beschwert, obwohl der Dealer ihm immer das richtige Share gesendet hat, sollte der Dealer diesen Wächter für Teilname an die Rekonstruktionsprotokolle sperren. Auf jeden Fall sollte die Reaktion der Wächer und des Dealer auf Beschwerden für die Wahrscheinlichkeit für einen Fehler bei der Verteilung der Shares abhängen.

Überprüfung der Konsistenz der Shares
Nachdem der Wächter W_i sein Share kontrolliert und als korrekt akzeptiert hat, muß die Gleichung für jedes Share gelten: $S_i = g^{si} \mod p$. Mit dieser Voraussetzung können die Wächter auch die Konsistenz ihrer Shares überprüfen. Durch die Veröffentlichung der Werte $C_j = g^{c_j} \mod p$ für jeden Koeffizient des Polynom $f(x)$ hat der Dealer auf dieses Polynom und insbesondere durch $C_0 = g^{c_0} \mod p$ auf den Wert des Schlüssels K festgelegt. Jeder der Wächter kann überprüfen, ob sein Sahre s_i der Wert des Polynoms $f(x)$ an der Stützstelle x_i ist, indem er $\prod_{j=0}^{k-1} C_j^{x^j} \mod p = g^{c_0 + c_1 x_i^1 + \cdots, c_{k-1} x^{k-1}} \mod p$ berechnet und mit S_i vergleicht. Wenn der Wächter W_i das richtige Share $s_i = f(x_i) = c_0 + c_1 x_i^1 + \cdots, c_{k-1} x^{k-1}$ bekommen hat, muß $\prod_{j=0}^{k-1} C_j^{x^j} \mod p = g^{c_0 + c_1 x_i^1 + \cdots, c_{k-1} x^{k-1}} \mod p = g^{s_i} = S_i$ (nach dem Satz Zahlentheorie) gelten. Da die Werte x_h, S_h und C_j auch für andere Wächter öffentlich bekannt sind, kann jeder durch $\prod_{j=0}^{k-1} C_j^{x_h^j} \mod p = S_h$ überprüfen, ob der Werte $x_h = f(x_h)$ als Exponent in $S_h = g^{s_h}$ vorkommt. Wenn dies für alle Wächter bekannt ist und jeder Wächter sein Share kontrolliert und akzeptiert hat, kann der Wächter W_i sicher sein, daß die verbleibenden Wächter als Shares die Werte des Polynoms an den entsprechenden Stützstellen erhalten haben. Nach dem Inter-

polationstheorem kann das Polynom $f(x)$ mit diesen k Werten und damit auch den Schlüssel $K = c_0$ eindeutig bestimmt werden. Damit kann die Konsistenz der Shares sichergestellt werden, wenn die Gleichung $\prod_{j=0}^{k-1} C_j^{x_h^j} \mod p = S_h$ für die Werte x_h, S_h und C_j erfüllt ist.
Wenn jedoch der Dealer dem Wächter W_i als Share s_i einen falschen Wert gesendet hat, für den $S_i \neq y = f(x_i) = c_0 + c_1 x_i^1 + \cdots, c_{k-1}x^{k-1}$ gilt, könnte dies mit $\prod_{j=0}^{k-1} C_j^{x_i^j} \mod p = g^{c_0 + c_1 x_i^1 + \cdots, c_{k-1}x^{k-1}} \mod p = g^y \mod p \neq S_i$ von jedem entdeckt werden und man kan davon ausgehen, daß der Dealer und ehrlich ist.

Die Sicherheit des Schemas von Pedersen

Die Sicherheit des Schema von Pedersen beruht darauf, daß die Berechnung des diskreten Logarithmus in vernünftiger Zeit undurchführbar ist. Es handelt sich also um ein computational-sicheres Secret Sharing Schema.
Zur Untersuchung der Sicherheit dieses Schemas wird zunächst angenommen, daß der Spion *Mallory* die Shares von $(k \Leftrightarrow 1)$ Wächtern zu erfahren versucht. In diesem Fall folgt mit derselbeb Argumentation wie im (k, n)-Threshold-Schema von Shamir, daß Mallory für die Rekonstruktion den Wert eines weiteren Share berechnen muß und desweiteren in Abhängigkeit von seiner Wahl jedes jedes Element von $\mathbb{Z}_q$ als möglicher Schlüssel bekommt. Der Spion Mallory kann zwar auf dieser Weise den Schlüssel K offenbar nicht bestimmen. Das Schema von Pedersen gewährlestet jedoch keine perfekte Geheimhaltung (*perfect Secrecy*) für den Schlüssel. Der Grund dafür ist, daß Mallory aus dem öffentlich bekannten Wert $C_0 = g^{c_0} \mod p = g^K \mod p$ versuchen kann, den Schlüssel K mit einem bestimmten effizienten Algorithmus abzuleiten. Die (unbewiesene) kryptographische Annahme, auf welcher das Schema von Pedersen basiert, läßt wie folgt formulieren:
„Gegeben seien die Zahlen x, e und die große Primzahl p, für welche die Gleichung $x = K^e \mod p$ gilt. Dann gibt es keinen polynomiellen Zeitalgorithmus, mit dem der Wert vom K aus dem bekannten Werten x, e und p abzuleiten".
Mit dieser Annahme soll der Spion Mallory nicht in der Lage sein, den Schlüssel aus der Gleichung $C_0 = g^K \mod p$ abzuleiten. Demzufolge besteht in dem Protokoll von Pedersen auch durch einen Betrüger keine Gefahr, da alle Shares beim Aufdecken in der Rekonstruktionsphase anhand der veröffentlichten Werte $S_i = g^{s_i} \mod p$ kontrolliert werden können und damit jeden Betrug entdeckt und der Betrüger identifiziert werden kann. Damit ein Betrüger den Schlüssel bei seinem Betrug, auch wenn dieser mit hoher Wahrscheinlichkeit entdeckt wird, nicht in Erfahrung bringt, kann die Variante der Methode von Tompa und Woll (Seite 40) verwendet werden.

Implementierungsbeispiel

Das folgende Beispiel veranschaulicht, wie die Wächter ihre Shares anhand des Protokolls von Pedersen überprüfen können. Zur Demontration werden wir die Monome $\{1, x, x^2, \ldots, x^{n-1}\}$ als Basis, wie im (k, n)-Threshold-Schema von Shamir, verwenden, da sowohl die Shares-Erzeugung als auch Schlüssel-Rekonstruktion identisch verlaufen.

Die Shares zu dem Schlüssel $K = 3$ werden nach dem Schema von Pedersen auf n=5 Wächter verteilt werden, von denen $k = 3$ Wächter ausreichen soll, d.h. die autorisierten Teilmengen sind Elemente der Basis $\mathcal{A}^0$ der Threshold-Zugriffsstruktur $\mathcal{A}$, um den Schlüssel K zu rekonstruieren.
Zuerst wählt der Dealer zwei Primzahlen $p = 47$ und $q = 23$, so daß die zugehörigen Restklassenkörper $\mathbb{Z}_{47}$ und $\mathbb{Z}_{23}$ fest gelegt werden. Weiter wählt er Die Zahl $g = 16$ als der Generator der zyklischen Untergruppe von $\mathbb{Z}_{47}$ mit den $q = 23$ Elementen:

$$\{1, 16, 21, 7, 18, 6, 2, 32, 42, 14, 36, 12, 4, 17, 37, 28, 25, 24, 8, 34, 27, 9, 3\}.$$

Zur Erzeugung der Shares wählt der Dealer zufällig ein Polynom f vom Grad $k \Leftrightarrow 1 = 2$ mit $f(x) = 3 + 5x + 7x^2$ $(K = 3)$. Für jeden Koeffizient c_j $(j = 0, 1, 2)$ berechnet Dealer die Werte C_j:
$C_0 = g^{c_0} \mod p = 16^3 \mod 47 = 7$, $C_1 = g^{c_1} \mod p = 16^5 \mod 47 = 6$, $C_2 = g^{c_2} \mod p = 16^7 \mod 47 = 32$.
Danach berechnet der Dealer berechnet die Shares als Werte des Polynom in $\mathbb{Z}_{23}$ an den Stützstellen x_i: $\{x_1 = 4,\ x_2 = 14,\ x_3 = 11,\ x_4 = 6,\ x_5 = 9\}$, die er für jeden Wächter W_i zufällig aus $\mathbb{Z}_{23}$ wählt. Als Shares ergeben sich die folgenden Werten:
$s_1 = f(4) = (3 + 20 + 112) \mod 23 = 20$, $s_2 = 19$, $s_3 = 819$, $s_4 = 9$, $s_5 = 17$.
Anhand dieser Shares s_i berechnet Dealer für jeden Wächter W_i den Wert $S_i = g^{s_i} \mod p$. Es ergibt sich dann: $S_1 = 27$, $S_2 = 34$, $S_3 = 42$, $S_4 = 14$, $S_5 = 24$. Nun veröffentlicht Dealer die Werte: $p = 47$, $q = 23$ $g = 16$ $C_0 = 7$ $C_1 = 6$ $C_2 = 32$ und zu jedem Wächter W_i die Werte s_i und S_i und sendet *geheim* an jeden Wächter W_i als Shares s_i. Die Verteilungsphase ist damit abgeschlossen.

Für die *Überprüfung der Shares auf ihre Korrektheit* berechnet jeder der Wächter W_i zuerst $g^{s_i} \mod p$ und vergleicht das Ergebnis mit S_i. Dies geschieht wie folgt:

W_1: $g^{s_1} \mod p = 16^{20} \mod 27 = 27 = S_1$ W_2: $g^{s_1} \mod p = 16^{19} \mod 27 = 34 = S_2$
W_3: $g^{s_3} \mod p = 16^8 \mod 27 = 42 = S_3$ W_4: $g^{s_4} \mod p = 16^9 \mod 27 = 14 = S_4$

Das modifizierte (k, n)-threshold-Schema von M. Stadler

Mit dem Protokoll *verify-share* von Pedersen kann sich jeder Wächter W_i überzeugen kann, daß er ein korrektes Share vom Dealer erhalten hat und daß sein Share mit den der verbleibenden Wächter in einer für die Rekontruktion des Schlüssels autorisierten Teilmenge in der vorgegebenen Zugriffsstruktur konsistent ist. Mit diesem Protokoll kann der Wächter einem Prüfer, einer öffentlichen Dritt-Instanz, gegenüber allerdings nicht nachweisen, daß er tatsächlich ein konsistentes Share kennt, ohne daß Share selbst aufzudecken. M. Stadler (Literatur schreiben) erweiertete das Schema von Pedersen. Das Schema von Pedersen wird dafür durch ein Protokoll zwischen einem Wächter W_i und einem Prüfer P erweitert, mit dem der Wächter den Prüfer davon überzeugen kann, daß er zu dem veröffentlichen Wert S_i einen Wert s_i kennt, für den die Gleichung $S_i = g^{s_i} \mod p$ gilt. Dabei soll der Prüfer P den Wert s_i nicht erfahren. Für einen solchen Beweis wird in diesem Protokoll das Public-Schlüssel-

Kryptosystem von ElGamal benutzt, das im Kapitel 2 eingeführt wurde. Für den Einsatz dieses Kryptosystem im Pedersen-Schema muß der Dealer im Schritt 1 des Protokoll *share* die beiden Zahlen $g, h \in \mathbb{Z}_p$ veröffentlichen, die jeweils eine zyklixche Untergruppe G_g und G_h in $\mathbb{Z}_p$ mit q verschiedenen Elementen $\{1, g, \ldots, g^{q-1}\}$ und $\{1, h, \ldots, h^{q-1}\}$ generieren. Jeder Wächter W_i muß außerdem einen privat Schlüssel $d_i \in \mathbb{Z}_q$ und den passenden öffentlichen Schlüssel $e_i = hd_i \mod p$ veröffentlichen. Dann wird das Schema von Pedersen um das folgende Protokoll erweitert.

Das Protokoll *public-verify-share*

Der Wächter W_i und der Prüfer P wiederholt $K \Leftrightarrow mal$:

1. DerWächter W_i veröffentlicht mit $(A, B) = (h^\alpha \mod p\ ,\ e_i^\alpha s_i \Leftrightarrow 1 \mod q)$ die Elgamal-Verschlüsselung des Share s_i mit mit einer beliebigen $\alpha \in \mathbb{Z}_q$. Der Wächter W_i behauptet den Prüfer P gegenüber, daß $S_i = g^{s_i} \mod p$ für den Öffentlichen Wert S_i gilt.

2. Der Wächter W_i wählt zufällig ein $w \in \mathbb{Z}_q$ und berechnet die Werte $t_h = h^w \mod p$, $t_g = g^{e_i^w} \mod p$. Dann sendet er dem Prüfer P die beiden Werte t_h und t_g.

3. Der Prüfer sendet dem Wächter W_i ein zufällig gewähltes Bit b.

4. Der Wächter sendet dem Prüfer $r = w \Leftrightarrow b\alpha \mod q$.

5. Der Prüfer P kontrolliert für:

   (a) $b = 0$, ob $t_h = h^r \mod p$, $t_g = g^{e_i^r}$ und

   (b) $b = 1$, ob $t_h = h^r A \mod p$, $t_g = S_i^{Be_i^r} \mod p$ gilt.

Wenn diese Gleichugen für jeder der l Runden erfüllt sind, akzeptiert der PRüfer P als Beweis dafür, daß der Wächter W_i zu dem veröffentlicheten Wert S_i ein Share s_i mit $S_i = g^{s_i} \mod p$ kennt.
Die Konsistez der Shares kann der Prüfer genauso wie die Wächter (Seite 47) kontollieren.

Die Gültigkeit dieses Protokolls beruht darauf, daß der Wächter W_i zu dem veröffentlichten Wert S_i ein Share s_i mit $S_i = gs_i \mod p$ kennt, folgt aus der Tatsache, daß für die Elgamal-Verschlüsselung $(A, B) = (h^\alpha \mod p\ ,\ e_i^\alpha s_i \Leftrightarrow 1 \mod q)$ von s_i mit dem beliebigen $\alpha \in \mathbb{Z}_q$ und dem öffentlichen Schlüssel e_i von W_i gilt:

$$S_i^B \mod p = g^{s_i B} \mod p = g^{s_i s_i^{-1} e_i^\alpha} \mod p = g^{e_i^\alpha} \mod p.$$

Für diese Gleichung muß der Wächter W_i im Schritt 2 des Protokolls mit $t_g = h^w \mod p$ einen entsprechenden Vergleichswwert bestimmen. Mit zweiten Wert $t_h = h^w \mod p$ kann bezüglich $A = h^\alpha \mod p$ der Wert $r = w \Leftrightarrow b\alpha \mod q$ kontrooliert werden. Wenn der Wächter W_i für den öffentliche Wert S_i das Share s_i mit $S_i = g^{s_i} \mod p$ nicht kennt und denoch versuchen will, den Prüfer P davon zu überzeugen,

muß er dann die gefälschten Werte A' und B' veröffentlichen. Im Schritt 2 des Protokolls muß er dann die ebenfalls gefälschten Werte $t'_h \neq h^w \mod p$ und $t'_g \neq g^{e_i^w} \mod p$ an den Prüfer P senden, welche $t'_h = h^r A' \mod p$, $t'_g = S_i^{B' e_i^r} \mod p$ für einen vorher von dem Wächter W_i gewählten Wert r erfüllen.
Wenn der Prüfer P im Schritt 3 an den Wächter W_i allerdings das Bit $b = 0$ sendet, müßte dieser für den Schritt 4 den Wert r' mit $t'_h = h^{r'} \mod p$ und $t'_g = g^{e_i^{r'}} \mod p$ berechnet. Aus der Annahme, daß die Berechnung des diskrette Logarithmus in $\mathbb{Z}_p$ in vernünftiger Zeit nicht durchgeführt werden kann, müßte der Betrug des Wächters W_i mißgelingen.
Wenn der Wächter W_i andererseits im Schritt 2 des Protokolls die beiden t'_h und t'_g deart bestimmen, daß die Gleichungen $t'_h = h^r \mod p$, $t'_g = g^{e_i^r} \mod p$ und der Prüfer im Schritt 3 das Bit $b = 1$ an den Wächter W_i sendet, müßte er für den Schritt 4 den Wert r' bestimmen mit $t_h = h^{r'} A \mod p$, $t_g = S_i^{B e_i^{r'}} \mod p$.Auch in diesem Fall scheitert der Wächter W_i wegen der Undurchführbarkeit der Berechnuung des diskreten Logarithmus an seinen Betrug.

Andererseits sind die Gleichungen im Schritt 5 des Protokolls erfült, wenn der Wächter W_i das Protokoll folgt und $(A, B) = (h^\alpha \mod p\ ,\ e_i^\alpha s_i^{-1} \mod q)$ gilt. Der Wächter W_i sendet an den Prüfer P im Schritt 2 die Werte $t_h = h^w \mod p$ und $t_g = g e_i^w \mod p$.
Wenn der Prüfer im Schritt 3 das Bit $b = 0$ bestimmt, sendet der Wächter ihm im Schritt 4 den Wert $r = w$, so daß die Gleichungen $h^r \mod p = t_h = h^w \mod p$ und $g^{e_i^r} \mod p = t_g = g^{e_i^w}$ im Schritt 5 offenbar erfüllt sind.
Wenn der Prüfer im Schritt 3 das Bit $b = 1$ bestimmt, dann sendet der Wächter ihm im Schritt 4 den Wert $r = w \Leftrightarrow \alpha \mod q$ mit dem Satz (aus Seite 44) erbigt sich

$$h_r^A \mod p = h^{w-\alpha} h^\alpha \mod p = h^{w-\alpha+\alpha} \mod p = h^w \mod p = t_h \quad \text{und}$$
$$S_i^{B e_i^r} \mod p = S_i^{e_i^\alpha s_i^{-1} e_i^{w-\alpha}} \mod p = g^{s_i s_i^{-1} e_i^w} \mod p = t_g.$$

Zum Abschluß des Kapitel 4 wird ein Implementierungsbeispiel präsentiert, welches bestätigt, wie das Protokoll *public-verrify-share* von M. Stadler arbeitet. Dabei will der Wächter W_3 aus dem Beispiel zum *„Distributed Provers with Applications to Undeniable Signatures* von Pedersen einem Prüfer davon über zeugen, daß er für für den öffentlichen Wert $S_3 = 42$ tatsächlich das Share s_i mit $S_3 = g^{s_3} \mod p$ kennt, ohne dieses Share dem Prüfer zu zeigen.
Zu Beginn des Protokolls werden die Werte $q = 23$, $p = 47$, $g = 16$ und $h = 6$ vom Dealer veröffentlicht. Mit dem Protokoll *share* sendet er an den Wächter W_3 das Share $s_3 = 8$ und veröffentlicht dazu den Wert $S_3 = g^{s_3} \mod p = 42$. Um dies einem Prüfer P beweisen zu können, wählt der Wächter W_3 seinen privaten Schlüssel $d_3 = 9$ und berechnet den zugehörigen öffentlichen Schlüssel $e_3 = h^{d_3} \mod p = 3$. Der Wächter W_3 und der Prüfer P folgen dem Protikoll *public-verify-share* in l Runden, von denen als Beispiel nur eine Runde vorgeführt wird.
Im Schritt 1 berechnet der Wächter W_3 die Elgamal-Verschlüsselung $(A, B) = (h^\alpha \mod p, e_i^\alpha s_i \Leftrightarrow 1 \mod q) = (6^4 \mod 47, 3^4 3 \mod 23) = (27, 13)$ mit einem Zahl $\alpha = 4 \in \mathbb{Z}_{23}$ und mit dem zu s_3 inversen Elemnt $s_3^{-1} = 3$ reduriert *modulo* 23. Der Wächter

W_3 veröffentlicht $(A, B) = (27, 13)$ und behauptet, daß es sich um die Elgamal-Verschlüsselung seines Share s_3 handelt, für welches $S_3 = g^{s_3} \mod p$ gilt. Dann wählt er im Schritt 2 eine zufällige Zahl $w = 7 \in \mathbb{Z}_{23}$ und berechnet die Werte $t_h = h^w \mod p = 6^7 \mod 47 = 4$ und $t_g = g^{e_3^w} \mod p = 16^{3^7} \mod 47 = 21$, welche er dann an den Prüfer sendet. Nun entscheidet sich der Prüfer mit dem Bit b im Schritt 3 über weiteren Ablauf dieser Runde.
Wir nehmen zunächst an, daß er das Bit $b = 0$ an den Wächter W_3 sendet, der ihm danach im Schritt 4 den Wert $r = w \mod q = 7$ als Antwort rücksendet. Damit berechnet der Prüfer die Werte $h^r \mod p = 6^7 \mod 47 = 4$ und $g^{e_3^r} \mod p = 16^{3^7} \mod 47 = 21$ und vergleicht die Ergebnisse mit $t_h = 4$ und $t_g = 21$. Falls die übereinstimmung vorliegt, ist es dann bewiesen, daß der Wächter W_3 in dieser Runde nicht versucht hat, den Prüfer mit gefälschten Werten $t'_h = h^r A' \mod p$ und $t'_g = S_3^{B' e_3^r} \mod p$ zu betrügen.
Wenn jedoch der Prüfer im Schritt 3 das Bit $b = 1$, erhält er im Schritt 4 als Antwort $r = w \Leftrightarrow \alpha \mod q = 7 \Leftrightarrow 4 = 3$ und im Schritt 5 mit den Rechnungen $h^r A \mod p = 6^3 27 \mod 47 = 4$ als Ergebnis, welches mit dem Wert $t_h = 4$ übereinstimmt und $S_3^{B' e_3^r} \mod p = 42^{13 \cdot 3^3} = 42^{351} \mod 47 = 21$ als Ergebnis, welches mit dem Wert $t_g = 21$ übereinstimmt. Mit diesem Resultat kann der Prüfer feststellen, daß der Wächter W_3 in dieser Runde keinen Betrug mit gefälschten Werten $t'_h = h^r \mod p$ und $t'_g = g^{e_3^r} \mod p$ für eine vorher gewählte Zahl r versucht hat.

Kapitel 5

Anwendung

In diesem Kapitel beschreiben wir eine Anwendung von Secret Sharing Schemes. Bei dieser Anwendung schlagen wir ein System vor, welches das *Key-Recovery-Problem* löst. In diesem Kapitel werden die Speicherungs- und Verwaltungsprobleme der Shares, die zentral im einem Schlüsselspeicher, statt lokal auf jedem Wächter, gespeichert sind.

Die Benutzer registrieren selbstätig ihre Schlüssel K in einen Schlüesselspeicher. Die registrierten Schlüssel kann allerdings weder von dessen Besitzer noch von nicht durch System festgelegten Personengruppen wieder ausgelesen, modifiziert und gelöscht werden. Der Schlüsselspeicher wird von n Wächtern kontrolliert, so daß nur bestimmte Teilmengen der n Wächter den eingefügten Schlüssel, nach Anforderung, rekonstruieren dürfen. Die Sicherheit dieses Systems beruht darauf, daß eine beliebige Wächtermenge, welche den Systembedingungen nicht genügt, den Schlüssel nicht rekonstruieren kann. Dies gilt auch für beliebige Angreifer.

Um das Problem unter Zuhilfenahme von Secret Sharing Schemes lösen zu können, formulieren das Problem Unter Verwendung des (k, n)-Threshold Schemas wie folgt:

Bei der Registrierung des Schlüssel (Keys) eines Benutzers mit dem (k, n)- Threshold Schema wird der Schlüssel vertrauenswürdigen Instanz (Dealer) in n Stücke geteilt. Die Stücke (*Shares, Shadows*) werdem jeweils *geheim* auf die Wächter (*Participants, Guards oder Trustees*) verteil. Statt jeder Wächter sein Stück als private Partialinformation lokal behält, werden die Shares in einem Schlüsselspeicher (*Key Storer*) gehalten, welchen von n Wächtern kontrolliert werden, so daß (k) oder mehr Wächter den Schlüssel nach Aufforderung rekonstruieren können. $(k \Leftrightarrow 1)$ Wächter oder beliebige Angreifer können keine Information über den Key erhalten. Eine wichtige Anforderung an den Schlüsselspeicher ist, daß die darin gehaltenen Shares die Rekonstruktionsbedingung des (k, n)-Threshold Schemas erfüllen muüssen. Sind die Shares nicht *korrekt* bzw. *korrupt*, so wird der falsche Schlüssel rekonstruiert bzw. kann die Rekonstruktion des Schlüssels nicht durchgeführt werden. Außerdem muß der Wächter seinen richtigen Share für die Rekonstruktion des Schlüssels bekommen. Mit anderen Worten: Die Korrekheit und Konsistenz der Shares im Schlüsselspeicher müßen gewährleistet werden.

Weshalb der Schlüsselspeicher eine wichtige Komponente bei unserer Anwendung sein soll, ist die Tatsache, daß durch Auswahl einer Zugriffsstruktur die Menge der Shares eines Wächters sehr groß werden kann. Im Kapitel 3 und 4 wurde gezeigt, daß die Sicherheit des Secret Sharing Schemas nimmt mit wachsender Anzahl der Shares eines Wächters und damit auch die Gesamtheit der Shares des Systems ab. Ein weiterer Grund dafür ist, daß in traditionellen Secret Sharing die Shares jedes Wächter lokal gespeichert werden sollen, da er für seine Shares allein verantwortlich ist und mit seinen Shares an der Rekonstruktion des Schlüssels beteiligen kann. In der Praxis arbeitet das System aber nicht mit einem Benutzer, dessen Schlüssel mit Hilfe des Secret Sharing Schemas geschützt werden soll, sondern eine Menge von Benutzern, die das gleiche Interesse haben. Es scheint zunächst gegensätzlich zu der Definition von Secret Scharing Schemes zu sein, die wir im Kapitel 4 kennengelernt haben. Doch haben wir im Kapitel 4 beschrieben, daß selbst die hier verwendeten Secret Sharing Schemes modifiziert werden müssen, um einige Probleme in realistischen Anwendungen lösen zu können.

In dieser Arbeit ist es wichtig, wie kryptographische Schlüssel der Mitarbeiter in einer Firma effektiv geschützt werden können, so daß sie nach Anforderungen *redundant* rekonstruiert werden können. Um dieses Ziel zu erreichen, müssen wir mit einigen Teilprobleme, die bei der Anwendung des (k, n)-Threshold Schemas für unsere Anforderungen entstehen werden, beseitigen.

Wir wählen das (k, n)-Threshold-Schema, denn es einfach zu implementieren ist. Jeder Wächter, er an der Rekonstruktion beteilig ist hat einen Share, so daß wir die Lösung unseres Problems nicht nur einfach realisieren können, sondern auch speicherplatz-effizient. Durch ideale Eigenschaft des (k, n)-Threshold-Schemas tritt im Vergleich mit perfekten Secret Sharing Schemata kein Kommunikation-Overhead auf. Allerdings hat dieses Verfahren einige Sicherheitslücken, die wir schließen müssen, bevor wir es anwenden. Daraus ergibt sich die folgenden Fragestellung, welche die Anforderungen in diesem Kapitel darstellen:

5.1 Anwendungsprobleme

Wie können die zu registrierenden Schlüssel von Benutzern sicher zu den Schlüsselspeicher gelangen?. Diese Frage wird in dem (k, n)-Threshold-Schema von Shamir nicht beantwortet. In dem Schema wird allerdings vorausgesetzt, daß die Shares nach ihrer Erzeugung sicher und geheim auf die Wächter verteilt werden. Diese Voraussetzung ist für praktische Anwendungen ist zu streng, denn die Übermittlung der Shares über ein Kommunikationskanal beispielsweie im LAN oder Intranet kann verursachen, daß die Shares durch Übertragunfehler verfälscht und angegriffen werden. Die Wächter wissen daher nicht, ob ihre Shares für die Rekonstruktion gültig sind. Ein Share heißt nicht gültig, wenn es entweder nicht korrekt oder mit Fehler behaftet ist. Ein ungütiges kann auch bedeuten, daß es von irgendeinem Angreifer „von außen“ stammen, da er die Rekonstruktion des Schlüssels verhindern will. Daß die Wächter bei der Rekonstruktion korrekte Shares erhalten, ist eine der Voraussetzung für die

redundante Rekonstruktion, die jedes Secret Sharing Schema als Basiseigenschaft hat.

Wie kann die Fälschungssicherheit der Shares garantiert werden? In der Tat setzt das Schema von Shamir ebenfalls voraus, daß die Wächter ehrlich sind. In der Realität kann es vorkommen, daß ein Wächter beispielsweise behaupten könnte: er habe das Share nicht erhalten, sondern ein anderes. Dieses falsches Share kann er bei der Rekonstruktion ausgeben. Damit wird der Schlüssel falsch rekonstruiert.

Das nächste Problem ist, wie kann ein Wächter nach der Share-Verteilung überprüfen, daß er ein korrektes Share hat. Dies ist, wie oben beschrieben ist, eine wichtige Voraussetzung für eine korrekte Rekonstruktion des Schlüssels.

Da die Rekonstruktion des Schlüssels in einem Secret Scharing Schema nicht vom einzelnen Wächter durchgeführt wird, sondern auf verschiedene Wächter verteilt, welche zu einer autorisierten Teilmenge angehören, genügt es noch nicht, wenn ein Wächter mit Hilfe eines Prüfschemas weißt, daß sein Share korrekt ist. Dieser Wächter muß sicher gestellt werden, daß sein Share mit den Shares der verbleibenden Wächter, die ebenfalls einer autorisierten Teilmenge angehören, konsistent ist, so daß k Shares sicher ausreichen, um den Schlüssel zu rekonstruieren.

Secret Sharing Schema ist ein Multi-Partei-Protokoll, mit dem die widerstreitenden Wächter auch den Schlüssel rekonstruieren können. Diese Anforderung kann das (k, n)-Threshold-Schema von Schamir leider nicht erfüllen. Das Schema setzt voraus, daß die Wächter gegenseitig vertrauen müssen, um den Schlüssel zu rekonstruieren. Das Schema kann zwar den Schlüssel mit hoher Redundanz rekonstruieren, d.h. der Schlüssel kann immer noch rekonstruiert werden, wenn höchstens $(n \Leftrightarrow k)$ Shares korrupt werden, erlaubt aber bis zu $(k \Leftrightarrow 1)$ Betrüger. Bekanntlich ist das Schemas nicht sicher gegen Betrüger. In Anwendungen, bei denen der Schlüssel durch widerstreitende Wächter rekonstruiert werden soll, kann ein unherlicher Wächter die $(k \Leftrightarrow 1)$ ehrliche Wächter betrügen, damit er allein den Schlüssel rekonstruieren kann. Andererseit können $k \Leftrightarrow 1$ W2achter können den $k \Leftrightarrow ten$ Wächter betrügen, um entweder beim Aufdecken zu erfahren oder um die Rekonstruktion des Schlüssels zu verhindern,indem sie ihre falschen Shares für die Rekonstruktion freigeben. Die Redundanz der Rekonstruktion nimmt somit ab, wenn das Problem nicht behandelt wird.

Das Problem der Betrugsprobleme können immer noch nicht gelöst werden, wennn Betrugsversuchen durch ein Protokoll verhindert werden. Denn die Betüger können nicht identifiziert werden, damit sie bei der Rekonstruktion des Schlüssels ausgeschlossen werden. Dies ist für jede effiziente Implementierung eines Secret Sharing Schemas erforderlich, denn es ist nicht wünschenswert, daß das Protokoll andauernd abgebrochen wird, wenn Betugsversuchen entdeckt worden sind. Wie können die Betrüger bei ihren Betrudsversuchen identifiziert werde?

Obwohl Betrugsversuchen entdeckt, Betrüger identifiziert werden können, haben die Betrüger den Schlüssel. Wenn dies nicht verhindert wird, würde sich die Geheimhaltung der kryptographischen Schlüssel unter Zuhilfenahme von Secret Sharing Schemata keinen Sinn ergeben.

Die Hauptproblem hier sind: Wie kann der Schlüsselspeicher realisiert werden, so

daß die redundante Rekonstruktionseigenschaft des eingesetzt Secret Sharing Schemas nicht verletzt wird. Offenbar stellt jedes Secret Sharing Schema ein Protokoll zwischen Dealer und Wächtern. Die durch im Kapitel 4 beschriebenen Protolle in modifizierten (k, n)-Threshold-Schemata, beispielsweise von Pedersen, beansprucht bei Prüfvorgängen stets Interaktionen zwischen Dealer und Wächtern, damit die Korrektheit und Konsistenz ihrer Shares gewährleistet werden. Diese Interaktionen sind erforderlich für die Erhaltung der redundanten Rekonstruktion des Schlüssels.

Ein weiteres Problem ist die Einflüsse des Administrators, dessen Anwesenheit im realistischen Netzwerk unverzichtbar ist. Ein Administrator kann beispielsweise die Gesamtheit der für die Rekonstruktion verfügbaren Shares im Schlüsselspeicher sogar unbeabsichtigt löschen und Programme installieren, welche die Funktionalität von Share-Verteilung- und Key-Recovery-Prozessen beeinflüssen, so daß unvorhersehbare Wirkungen auf das gesamte System hervortreten können. In diesem Zusammenhang müssen einige Annahmen gemacht werden, um eine Toleranz für die Funktionalität des Systems zu erreichen.

Wichtig ist auch die Behandlung der Shares nach jeder der Rekonstruktion eines Schlüssels, denn es ist aufgrund der Sicherheit nicht wünschenswert, die für die Rekonstruktion benutzten Shares nochmals zu verwenden, da sie von den Wächtern in der autorisierten Teilmenge aufgedeckt werden. Das Problem ist: Wie können die Shares erneuert werden, ohne den Schlüssel zu verändern. Die einfache Methode war, das der Wert des Schlüssels nach seiner Rekonstruktion geändert werden soll. Diese änderung bedeutet auch, daß neue Verteilung der Shares vorgenommen werden müssen. Es hat zur Folge, daß Kommunikation-Overheads auftreten und damit auch hohe Kommunikationskosten verursacht werden.

Die Mehrheit von veröffentlichten Secret Sharing Schemata beschäftigen sich mit der Entwicklung von Protokollen für einen Schlüssel (Single Key Secret Sharing). Wenn mehr Schlüssel registriert werden, was in unserer Anwendung offenbar der Fall ist, ist es erforderlich, die Shares im Schlüsselspeicher aufgrund ihrer Speicherung und Zugriffsorganisation durch eine Konzeption zu organisieren. Es ist an dieser Stelle anzumerken, daß die Speicherung und Organisation der Shares ohne den eingeführten Schlüsselspeicher offensichtlich noch komplizierter werden, wenn die Gesamtheit der Shares aller eingefügten Schlüssel jeweils bei jedem der n Wächter lokal gespeichert werden müssen, wie bei meisten Secret Sharing Schemata ersichtlich ist.

Alle dieser genannten Probleme und deren „ausreichenden“ Lösungen wurden im Kapitel 4 diskutiert und ausführlich beschrieben. Ziel dieses Kapitels ist, die ausgearteten Ergebnisse zu benutzen, um ein Konzept für das Schlüssel-Recovery-System anhand eines Schlüsselspeichers angeben zu können. Dafür braucht man ein Modell, welches das System beschreiben kann. Mit den Resultaten aus dem Kapitel 4 wird das Konzept für die Funtionalität, inbesondere für den Schlüsselspeicher beschrieben.

5.2 Das Modell und Einschränkungen

5.2.1 Das Modell

Das Modell ist ein komplexes Abbild eines *Schlüssel-Rückgewinnung-Systems* (kurz: SRS). Das Modell wird zunächst durch die Abbildung 5.1 grob dargestellt werden. Im weiteren Sinn kann es aus folgenden Komponenten bestehen:

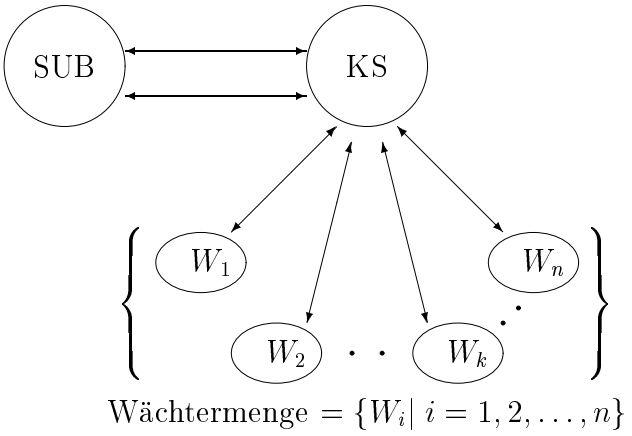

Abbildung 5.1: Ein Modell für das Schlüssel-Rückgewinnung-System (SRS).

- einer Menge von *Benutzern/Anwendern*, die ihre Schlüssel im SRS registrieren lassen.
- Einem *Submitter*, der jeden eingefügten Schlüssel in n Shares zerlegt und *Schlüsselspeicher* sendet.
- dem Schlüsselspeicher, die vom Submitter gesendeten n Shares anhand einer bestimmten *Speichertechnik* speichert.
- einer Menge von n Wächtern, von denen k Wächter ausreichen, die eingefügten Schlüssel nach Aufforderung anhand des (k, n)-Threshold-Schemas von Shamir rekonstruieren
- und verschiedenen *Kryptoeinrichtungen* sowie weitere Systemkomponenten, die im Verlauf dieses Kapitels schrittweise erläuchtert werden.

Im folgenden werden die einzelnen Komponenten und deren Zusammenspiel beschrieben, so daß sie als Grundlagen für die Konzepte eines RSR darstellen.

Submitter
Der Submitter hat die Roll eines Dealer in dem (k, n)-Threshold-Schema. In den

Submitter können die Benutzer ihre Schlüssel eingeben. Auf dem Submitter läuft der Prozess P_{Sub}, welcher das Protokoll *share* ausführt, so daß der einzelnen eingegebene Schlüssel mit dem n Shares zerlegt werden. Die Shares sendet der Submitter *sicher* und *verbindlich* an den Schlüsselspeicher. Danach kann er den nächsten Schlüssels auf der gleichen Weise annehmen.
Die Kommunikation zwischen Benutzern, Submitter und dem Schlüsselspeicher erfordert ein Kommunikationskanal größerer Bandbreite, denn die Registrierung findet eher statt als Rekonstruktion.

Schlüsselspeicher
Der Schlüsselspeicher fungiert als ein *Schlüssel-Server bzw. Datenbank-Server*, der die vom Submitter erzeugten n Shares für die Wächter aufnimmt. Die Anforderungen an den Schlüsselspeicher sind: Die dort vorliegenden Shares müssen korrekt und konsistent sein.
Auf dem Schlüsselspeicher läuft der Prozess p_{KS}, der die vom Submitter gesendeten Shares speichert. Außerdem werden die Shares im Schlüsselspeicher mittels einer Chiffriertransformation eines Public-Key-Kryptosystems verschlüsselt, so daß nur die Wächter die Shares mit ihren entsprechenden öffentlichen Schlüsseln bzw. geeigneten *Speichertechnik* für die Schlüsselrekonstruktion zugreifen dürfen. Damit der Zugriff auf die Shares möglich ist, sind der Schlüsselspeicher mit den Wächtern über ein Netzwerk mit niedrigerer Bandbreite verbunden, da die Schlüsselrekonstruktion im Vergleich mit Schlüsselregistrierung seltener stattfindet.

Die Wächter
Die Wächter sind Elemente der Menge $\mathcal{W} = \{W_i | 1 \leq i \leq n\}$, bezeichnet als Wächtermenge. Bezüglich des Rekonstruktionsvorganges werden sie auch als *Recovery Agents* genannt. Wird aufgefordert, daß ein bestimmter Schlüssel rekonstruiert werden soll, dann bilden die Wächter anhand der vorgegebenen Threshold-Zugriffsstruktur eine autorisierte Teilmenge mit k Wächtern. Diese k Wächter benutzen das Protokoll *recover* des eingesetzten (k, n)-Threshold-Schemas, um bestimmten Schlüssel eines Benutzers zu rekonstruieren. Die Wächter kontrollieren den Schlüsselspeicher
Die Wächter sollen bei jeder Schlüsselrekonstruktion miteinander kooperieren und gelegentlich zusammenarbeiten, um die Shares im Schlüsselspeicher zu *erneuern.*

Weiter gibt es einen *Sicherheits-Server*, der für Authentisierung der Benutzer dem System gegenüber veranwortlich ist. Anhang der Berechtigungsinformation ist es den Benutzern möglich, ihre zu registrierenden Schlüssel in den Submitter einzugeben. Einfachheit halber wird angenommen, daß der Sicherheit-Server auch die *Zertifikatenausgabe* übernimmt. Die Zertifikate aller Benutzer sind bei ihm gespeichert. Die Zertifikate werden durch den Sicherheit-Server ebenfalls signiert, so daß die Ersetzung der öffentlichen Schlüssel nicht möglich ist.
Die wichtigste Aufgabe des Sicherheits-Servers ist die Gefahr von *trajonische Pferden* zur Kenntnis zu nehmen und geeignete Maßnahmen zu ergreifen. Maschine, auf denen absolut keine trajonische Pferden stattfinden, sind Submitter, Schlüssel-Server bzw. Datenbank-Server sowie Sicherheits-Server.

Für das *zweite Konzept* kann der Schlüsselspeicher auch als *Datenbank-Server* fungie-

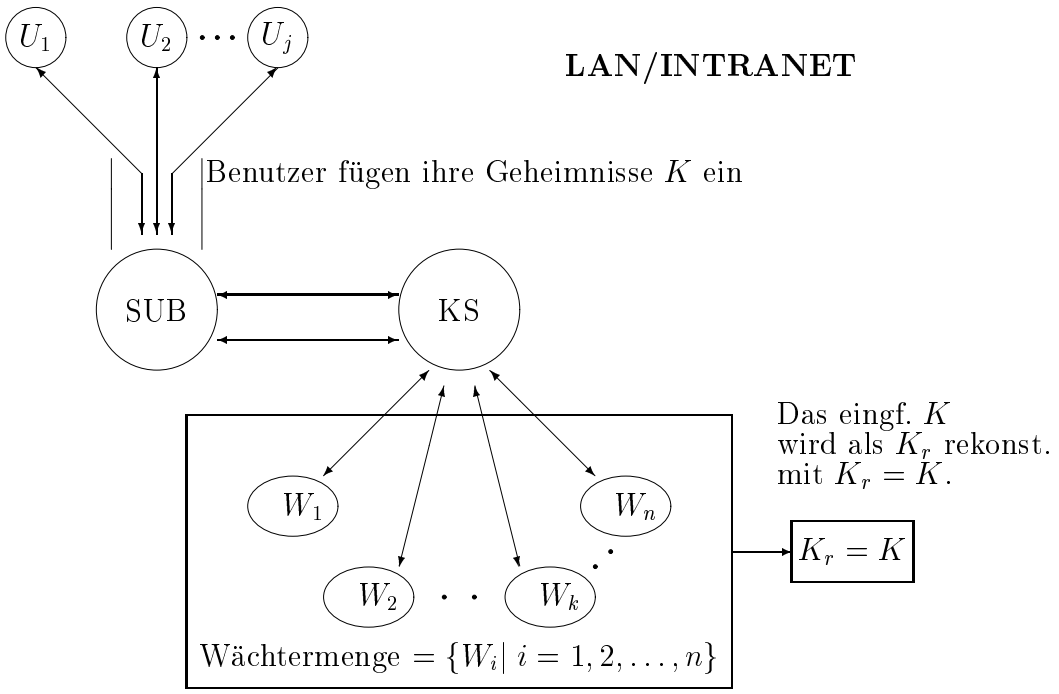

Abbildung 5.2: Das eingefügte Geheimnis K wird gemäß SRS rekonstruiert.

ren. Und die genannten Komponenten werden zu einer Einheit aufgrund verschiedener Konzeption zusammengefaßt. Allerdings gehen die bisher beschriebenen Funktionalität des SRS nicht verloren. Wir werden im Verlauf der Untersuchung noch dazu kommen.

Wie in meisten Systemen gibt es hier auch einen Administrator, der als *Sichersadministrator* bezeichnet wird. Seine Augaben sind: Die Systemsoftware zu installiert, das System pflegen, Betrugsversuchen dorkumentieren, Backup durchführen etc. Anhand dieses Modells werden neben *Geheimhaltung, Verbindlichkeit, Intergrität und Authetisierung* folgenden Anforderungen gestellt:

Durch dieses Modell können, wie in der Abbildung **??**. die folgenden „Endanforderungen“ für die Rekonstruktion erzielt werden:

1. **Umgebung**: Das SRS kann in einem *öffentlichen Netzwerk* plaziert werden. Die Anforderung wird erfüllt, indem die kryptographische Tools im Kapitel 2 geeignet eingesetzt werden. Neben Verschlüsselung und Signatur-Verfahren werden wir einige Protokolle vorschlagen.

2. **Funktionalität**: Der Benutzer kann seinen Schlüssel K ins System eingeben. Falls der eingefügte Schlüssel rekonstruiert werden soll, dann kann der eingefügten Schlüssel mit *hoher Redundanz* als Ausgabe K_r mit $K_r = K$ berechnet. Diese Rekonstruktionseigenschaft garantiert eingentlich jedes Secret

Sharing Schema. Allerdings Durch Angriffe und Inkonsistenz der Shares kann die Rekonstruktioneigenschaft stark reduziert werden. Dies werden wir durch Modifikation bestimmter vorhandenen kryptographischen Protokolle.

3. **Redundanz**: Die Redundanz gekennzeichnet sich durch die *Robusheit* des Systems, d.h. Die eingefügten Schlüssel K sollen auch im Vorhandensein der Angreifer sowie Korruption und Inkonsistenz der Shares rekonstruiert werden

4. **Sicherheit**: Das System soll einerseits, wie oben beschrieben ist, sicher gegen kryptographische Angriffe sein, wobei

   - es nicht manipuliert werden kann, so daß $K_r \neq K$ ist und
   - nicht vom System festgelegte Teilmenge der Wächte bzw. beliebige Angreifer nicht in *vernüntiger Zeit* den orginalen Schlüssel K in Erfahrung bringen können.

5. und andererseits eine eindeutige *Sicherheitseigenschaft* besitzen: berechnungssicher oder uneingeschränkt sicher.

Damit die Behandlung der gestellten Anforderungen im Zusammenhang stehen kann, ersetzen wir das genannten Modell durch den Begriff *Schlüssel-RU2ckgewinnung,system.*

5.2.2 Einschänkungen

Das Netzwerk
Sowohl der Submitter als auch der Schlüsselspeicher sollen ehrlich sein. Es ist ausgeschlossen, daß der Submitter während seiner Ausführung des Protokolls *share* von außen angegriff wird, so daß er unehrlich wird, d.h. falsche Shares werden erzeugt und weiter an den Schlüsselspeicher gesendet. Es ist ebenfalls ausgeschlossen, daß die Speicherung der Shares nach dem Empfang durch den Schlüsselspeicher durch Angriffe gehindert wird, so daß die falsche Shares im Schlüsselspeicher vorliegen.

Es ist anzunehmen, daß es fehlererkennende und -korrigierende Codes gibt, die die Übertragungsfehler bei der Benutzung des Kommunikationskanals zwischen den Komponenten des Modells korrigieren.

Jeder der Wächter ist an ein öffentlichen Netz (*Broadcast Network*) angeschlossen, d.h. die anderen Wächter, Schlüsselspeicher, Submitter und Angreifer hören mit, was gerade gesendet sind.

Jeder Kommunikationsteilnehmer verfügt über eine lokale Uhr, so daß er in der Lage ist, eigenen Zeitstempel (*timestamp*) zu erzeugen. Die Zeitsynchronisation im Netz wird durch Zeit-Server übernommen.

Es ist sinnvoll vorauszusetzen, daß die Benutzer, welche ihre Schlüssel registrieren wollen, über ein Breiband-Netz mit dem Submitter kommunizieren können. Diese Voraussetzung gilt ebenfalls für die Kommunikation zwischen dem Submitter und

dem Schlüsselspeicher, denn die Shares zu den Schlüsseln nach ihrer Generierung werden durch den Submitter sofort an den Schlüsselspeicher gesendet. Im Gegensatz dazu erfordert die Kommunikation zwischen den Wächtern und dem Schlüsselspeicher sowie unter Wächtern ein Netzwerk, dessen Bandbreite vergleichsweise gering sein kann.

Die bisher behandelte *Sicherheitpolitik* konzentiert sich auf die „innere Sichheit". Durch Verschlüsselung und Signierung können nur die Geheimhaltung und Authentisierung der Shares gesichert werden. Im Kapitel 4 wurde ausführlich behandelt, wie die Korrekheit und Konsistenz der Shares durch die vorgestellten Secret Sharing Schemata überprüft werden können. Die redundante Rekonstruktion der eingefügten Schlüssel kann durch diese Sicherheitsvorkehrung garantiert werden. In einem öffentlichen Netzwerk.

Angriffsmöglichkeiten

Bei Initialisierung des SRS wird k und n so gewählt, daß die Wahrscheinlichkeit von mehr als $n \Leftrightarrow k$ Ausfällen hinreichend klein wird. Für den Parameter k wählen wir $k = \lfloor \frac{n}{2} \rfloor$ (ein Beweis für diese Parametrisierung liegt vor, Seite 61 und 54), damit die Anzahl der k-elemtigen autorisierten Teilmengen maximal wird, so daß bei Ausfällen die Schlüssel K rekonstruieren können.

Ein durch Angriff beschädigter Wächter ist gekennzeichnet dadurch, daß es den Angreifern Vorteile ermöglicht, entweder den Schlüssel in Erfahrung zu bringen oder mehr als $n \Leftrightarrow k$ Shares zu vernichten, so daß die Rekonstruktion des Schlüssels nicht durchgeführt werden kann. Weiter wird vorausgesetzt, daß es keinen Unterschied gibt zwischen den durch gegnerische Angriffe hervortretenden Beschädigungen der Wächter und den, die zum Beispiel durch Systemfehler hervorgerufen werden.

Der Gegner dürfen nicht die auf die Swap-Datei zugreifen, in der eventuell Daten während der Share-Erzeugung und Schlüssel-Rekonstruktion (UNIX) ausgelagert sind. Manipulation von Prozessen, insbesondere *share*- bzw. *recover*-, sowie System-I/O. sind ausgeschlossen. Der System Administrator sollte alle werdenden Missverhältnisse der Systemkomponenten auführlich dokumentieren, damit im Ernstfall Gegenmaßnahme rechzeitig ergriffen werden können.

Der Gegner sollen ständig im Netz und an der Kommunikation beteiligt sein. Sie können alle gesendeten Nachrichten mitbekommen. Allerdings können sie nicht unbeschädigte Wächter verhindern, daß sie die Nachrichten im Netz empfangen bzw. senden. Alle Algorithmen, die auf jedem Wächter ausgeführt werden, und nicht geschützte Daten sind den Angreifern bekannt.

Die Angreifer sollen nur über beschänkte Rechenkapazität verfügen. Sie kann daher einige im System verwendeten kryptographischen Primitiven in vernünftiger Zeit nicht brechen. Damit ist die Sicherheit des Systems zu der verwendeteten kryptographischen Primitiven äquivalent anzunehmen.

5.3 Modifikation

Damit Konzepte für das Schlüssel-Rückgewinnung-System ausbeartet werden, muß das (k, n)-thresld-schema von Shamir geeignet modifiziert werden:

- Der Dealer ist der Submitter. Submitter wird zunächte ein Rechner sein. Am Ende des Abschnittes diskutieren wir den Fall, daß jeder Benutzer ein Submitter ist.
- Das Schema von Shamir wird durch beide kryptographische Primitiven $ENC(\cdot)$ und $SIG(\cdot)$ erweitert, daß die Geheimhaltung, Verbindlichkeit und Integrität der Shares im offenen Netz gewährleistet werden.
- Mit dem Protokoll *authorisation-check*(A, B) kann die Benutzerauthentisierung realisiert werden, bevor Kommunikationsverbindungen aufgebaut werden.
- Mit dem Protokoll *refresh* werden die Shares wegen Schutz vor passiven Angreifern aktualiert. Dieses Protokoll ist erforderlich, da die Gesamtheit der Shares nicht nach der Zerlegung der zugehörugen Schlüssel durch dem Protokoll *share* nicht an die Wächter verteilt und dort gespeichert, sondern *zentral* im Schlüsselspeicher festgehalten.
- Obwohl das Schema von Shamir durch geeignete Wahl von k und n robust ist, d.h. die Rekonstruktion eines Schlüssels kann durchgeführt werden, wenn höchstens $n \Leftrightarrow k$ Shares korrupt werden. Mit dem Protokoll *recover-share* können die korrupten Shares identifiziert werden und entsprechend zurückgewonnen (*recover*) werden.

Diese Modifikationen werden im Verlauf des Abschnittes, um aussagekräftige Konzeptionen des SRS zu ermöglichen.
Die wichtigsten Komponenten im SRS, wie erwaähnt, der Submitter, als Implementierung Schlüssel-Server bzw. Datenbank-Server Wächter. Auf sie laufen alle wichtigen Prozesse des Secret Sharing Schemas. Es ist für sichere Rekonstruktion wünschenswert, daß die beiden Algorithmen zu signieren (In JAVA möglich).

Die Algorithmen *share*, *recover*, *refresh* und *recover-share* müssen daher von dem *TA Trusted Authority* signiert werden, um gegen *trajonische Pferden*. Da sind fremde, erst nützliche Programme, die von Systembenutzer/-anwender aufgrund seiner niedrigen Kosten bzw. freier Verteilung installieren. Dieser Fall ist in sofern gefährlich, *wenn Submitter selbst Benutzer sind*, denn ein Benutzer kann eigne Programme selbst installieren und damit auch „diese frenden Programme." In der ersten Modifikation wurden die beiden kryptographischen Primitiven $ENC(\cdot)$ und $SIG(\cdot)$ in SRS eingebettet. Sie werden wie folgt eingesetzt:

Die Verschlüsselung der Nachricht m wird mit $m' = ENC_{e_r}(m)$ bezeichnet, wobei e_r der Public-Key des Empfängers ist. Der Empfänger entschlüsselt die Nachricht mit seinem Privat-Key d_r, also $ENC_{d_r}(m') = ENC_{d_r}\big(ENC_{e_r}(m)\big) = m$.

Sendet der Wächter W_i eine Nachricht m an den Wächter W_j für $i \neq j$, dann besteht die gesendete Nachricht aus $\{m', SIG_{d_{W_i}}(m')\}$. Der Wächter W_j prüft die Echtheit der Nachricht mit dem Public-Key e_{w_i} von W_i und mit seinem Privat-Key d_{w_j} kann W_j die Nachricht m lesen.
Unter Verwendung des digitalen Signature-Standard (DSS) signiert der Wächter W_i insbesondere nur auf den Hashwert der Nachricht m und verschlüsselt m mit dem Public-Key e_{wj} des Wächters W_j. Der signierte Hashwert wird zusammen mit der verschlüsselten Nachricht an W_j verschickt. Demnach erhält W_j als Nachricht

$$m' = ENC_{e_B} \text{ und } y = SIG_{d_B}\big(H(m)\big).$$

W_j entschlüsselt die Nachricht mit $ENC_{d_{wj}}(ENC_{e_{wj}}(m') = m$ und wendet die Hashfunktion H auf m an, um den Hashwert $h' = H(m)$ zu berechnen. Anhand des öffentliche Verrifikationsalgorithmus $VER(\cdot)$ prüft W_j die Echtheit der Nachricht m:

$$VER_{e_A}(y) = ja, \text{ falls } h' = H(m).$$

Spezifikation für die Hashfunktion wurde im Kapitel 2 im Zusammenhang mit dem Birthday-Angriff ausführlich beschrieben.

5.4 Infrastruktur

Die Umgebung, in der das SRS untersucht wird, ist ein LAN, das aus verschlieden *Domänen* bestehen. Diese Domänen sind über einem Netzwerk miteinader verbunden.
Die Infrastruktur besteht aus zwei Teilnetzen, die zu einem LAN (*Local Area Network*) gehören. In einem Teilnetz existieren verschiedene Domänen, zu denen die Benutzer angehören. Damit die Benutzer ihre Schlüssel ins SRS registrieren dürfen, müssen sie eine *Zugangsberechtigung* zu SRS vom Sicherheitsadministrator haben. Die Berechtigung setzt sich aus:

1. dem *Benutzername* (, zu dem die eindeutige *Benutzeridentität*, in vom SRS erstellt und dort abgesiechert ist

2. einem Identifikationsschema (z.B. Password-Methode nach Lamport)zur Authentisierung mit dem Submitter.

Ein Benutzer kann verschiedene Benutzernamen haben, da er zu verschieden Domänen gehören kann. In dem gesamten Netzwerk wird er unter seiner eindeutigen Identität identifiziert.

5.5 Konzeption und Realisierung des Schlüsselspeichers

5.5.1 Bechreibung des Schlüsselspeichers

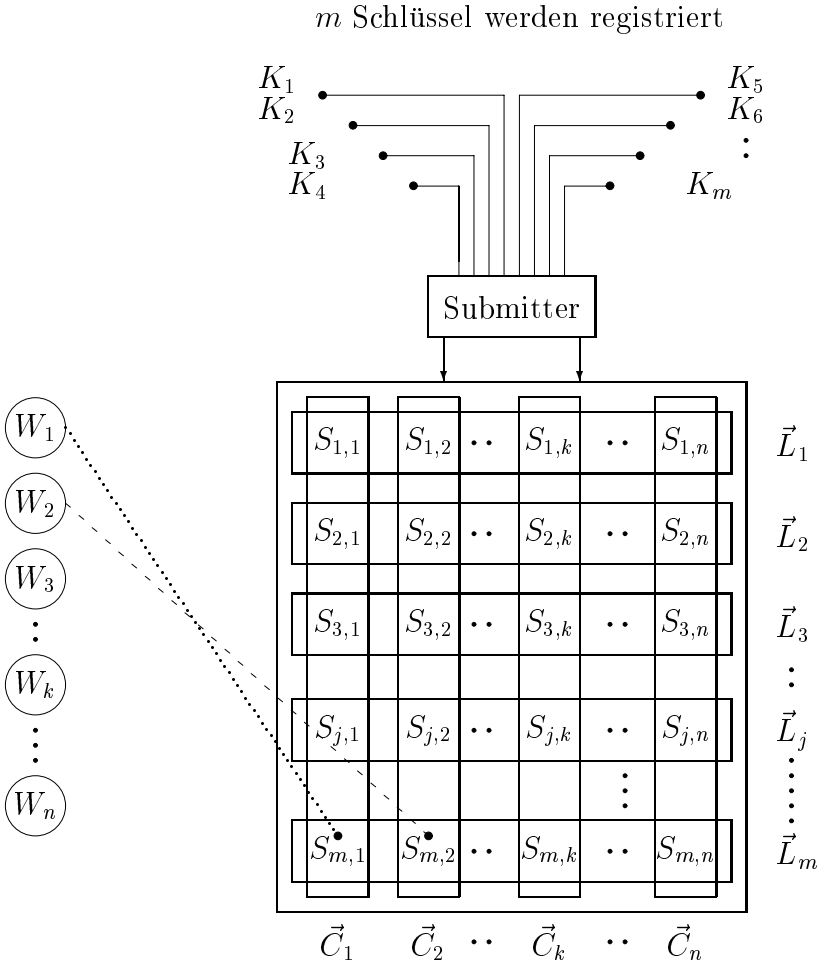

Abbildung 5.3: Realisierung des Schlüsselspeichers

Es sei $\mathcal{W} = \{W_1, \ldots, W_n\}$ die Wächtermenge mit (festen) n Elementen. $\mathcal{K}$ sei der Schlüsselraum. Wir nehmen zunächst an, daß jeder der m Benutzer U_j einen Schlüssel $K_j \in \mathcal{K}$ $(1 \leq j \leq m)$ in den Schlüsselspeicher registrieren will. Es ist sinnvoll anzunehmen, daß $m \gg n$ gilt.
Gemäß des (k, n)-Threshold-Schemas von Shamir wählt der Submitter zufällig für jeden U_j ein Polynom $f_j(x) \in \mathbb{Z}_p[x]$, wobei p prim ist und $p \geq n + 1$ gilt, vom Grad k mit $f_j(x) = a_{0,j} + a_{1,j}x + \cdots + a_{k-1,j}x^{k-1}$, $0 \neq a_{k-1,j} \in \mathbb{Z}_p = \mathcal{K}$ und $K_j = a_{0,j}$.
Mit dem Protokoll *share* generiert der Submitter $m \times n$ Shares zu m Schlüsseln K_j, indem er zunächst für ein $j \in \{1, 2, \ldots, m\}$ zufällig die n Stützstellen

$x_{j,1}, x_{j,2}, \ldots, x_{j,k}, \ldots, x_{j,n}$ aus $\mathbb{Z}_p$ wählt und dann die Shares für die n Wachter W_i wie folgt berechnet:

$$\begin{array}{c|cccccc} j=1 & f_1(x_{1,1})=S_{1,1} & \cdots & f_1(x_{1,k})=S_{1,k} & \cdots & f_1(x_{1,n})=S_{1,n}. \\ j=2 & f_2(x_{2,1})=S_{1,1} & \cdots & f_2(x_{2,k})=S_{2,k} & \cdots & f_2(x_{2,n})=S_{2,n}. \\ & \vdots & \vdots & \vdots & \vdots & \vdots \\ j=m & f_m(x_{m,1})=S_{m,1} & \cdots & f_m(x_{m,k})=S_{m,k} & \cdots & f_m(x_{m,n})=S_{m,n}. \end{array}$$

Einfachheit halber nimmt man an, daß der Submitter nach Generierung die Shares $S_{j,i}$ $(1 \leq j \leq m,\ 1 \leq i \leq n)$ an den Schlüsselspeicher senden kann, so daß ihre Integrität und Konsistenz gesichert sind, d.h. für jedes $j \in \{1, 2, \ldots, m\}$ rekonstruieren beispielsweise die Shares $S_{j,1}, S_{j,2}, \ldots, S_{j,k}$ anhand des (k, n)-Threshold-Schemas von Shamir den Schlüssel K_j. Dann kann der Schlüsselspeicher durch die $m \times n$ Matix $\mathbf{M}$ dargestellt werden:

$$\mathbf{M} = \begin{pmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,k} & \cdots & S_{1,n} \\ S_{2,1} & S_{2,2} & \cdots & S_{2,k} & \cdots & S_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{j,1} & S_{j,2} & \cdots & S_{j,k} & \cdots & S_{j,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{m,1} & S_{m,2} & \cdots & S_{m,k} & \cdots & S_{m,n} \end{pmatrix}$$

In dieser Matrix ist der Zeilenindex j gleich dem des Schlüssel K_j und der Spaltenindex dem des Wächter W_i. Weiter bezeichnen wir

$$\begin{aligned} \vec{L}_j &:= (S_{j,1}, S_{j,2}, \ldots, S_{j,k}, \ldots, S_{j,n} \text{ für } 1 \leq j \leq m\ , \\ \vec{C}_i &:= (S_{1,i}, S_{2,i}, \ldots, S_{j,i}, \ldots, S_{m,i})^T \text{ für } 1 \leq i \leq n \end{aligned}$$

als Rekonstruktionsvektor des (k, n)-Threshold-Schemas bzw. Privat-Vektor des Wächters W_i. Bei jeder Registrierung bleibt die Anzahl der Komponenten des Vektors $\vec{L}_j$ fest, während die Länge des Vektors $\vec{C}_i$ komponentenweise zunimmt. Ein Matrix-Element $S_{j,i}$ ist das Share des Wächters W_i zu dem eingefügten Schlüssel K_j für $1 \leq j \leq m$ und $1 \leq i \leq n$. In der Abbildung 5.3 wird die Registrierungsvorgang sowie die Speicherung der Shres $S_{j,i}$ zu den m Schlüsseln K_j dargestellt.

5.5.2 Sicherheit des Schlüsselspeichers

Um die „innere“ Sicherheit des Schlüsselspeichers auf kryptographischem Aspekt zu untersuchen, unterscheiden wir uns zwei Fälle:

1. Fall
Bezüglich der Matrix $\mathbf{M}$ hat der Wächter W_i vollen Zugriff auf den Spaltenvektor $\vec{C}_i$, da dessen Komponenten $S_{1,i}, S_{2,i}, \ldots, S_{j,i}, \ldots, S_{m,i}$ die Shares zu den jeweiligen Schlüsseln K_j $(1 \leq j \leq m)$ sind. Die folgende Proposition garantiert, daß sogar $k \Leftrightarrow 1$ Wächter mit ihren Privat-Vektoren keinen Schlüssel K_j rekonstruieren können.

Proposition 5.1. Es sei $W = \{W_1, \ldots, W_n\}$ die Wächtermenge und $\{S_{j,i} | \, 1 \leq j \leq m \text{ und } 1 \leq i \leq n\}$ seien die anhand des (k,n)-Threshold-Schemas vom Submitter berechneten Shares zu den Schlüsseln K_j. ($k \Leftrightarrow 1$) Wächter können mit ihren Privat-Vektoren $\vec{C}_i$ ($1 \leq i \leq n$) keinen der m Schlüsseln K_j rekonstruieren.

Beweis. Dazu sei $\mathcal{A} = \{B \subseteq W : |B| \geq k\}$ die Threshold-Zugriffsstruktur. Für ($k \Leftrightarrow 1$) Wächter W_i gilt $W_i \in \{W_i\}$. Setze $B = \bigcup_{l=1}^{k-1} \{W_l\}$. Da $|B| = k \Leftrightarrow 1 < k$ ist, gilt $B \notin \mathcal{A}$. Die Menge B ist eine unauthorisierte Teilmenge. Die Wächter in B erhalten keine Information über die Schlüssel K_j.

Die Sicherheit des Schlüsselspeichers basiert in diesem Fall auf der uneingeschänkten des (k,n)-Threshold-Schema von Shamir.

2. Fall

Da die Shares $S_{j,i}$ im Schlüsselspeicher kryptographisch nicht geschützt sind, so kann beispielsweise ein beliebiger Wächter bzw. der Schlüsselspeicher selbst die Shares aufdecken und die Schlüssel K_j rekonstruieren. Betrugsversuchen können „im innen" stattfinden, allerdings nicht im Sinne der Untersuchungen im Kapitel 4. Es ist daher notwendig, den Schlüsselspeicher unter Zuhilfenahme der im Kapitel 2 diskutierten Public-Key-Kryptosysteme (RSA, Elgamal) Verschlüsselung und Signaturen (DSS) vor Angriffen zu schützen.

Wir nehmen an, daß der Wächter W_i nun die ($k \Leftrightarrow 1$) Wächter betrügen will. Dann muß er die $k \Leftrightarrow 1$ Spaltenvektoren $\vec{C}_1, \ldots, \vec{C}_{i-1}, \vec{C}_{i+1}, \ldots, \vec{C}_k$ im Schlüsselspeicher zugreifen, um alle m Schlüssel mit dem (k,n)-Threshold-Schema von Shamir rekonstruieren zu können. Um gegen diesen Angriff abzuwehren, wird die Verschlüsselungsfunktion $ENC_{e_{w_i}}(\cdot)$ eingesetzt. Das heißt, wir verschlüsseln die Matrix-Elementen $S_{j,i}$ mit dem Public-Key e_{w_i} des jeweiligen Wächters W_i. Wir definiert eine neue Matrix $\mathbf{M}_E$, deren Elementen gleich den durch die Verschlüsselungstranformation $ENC_{e_{w_i}}(\cdot)$ entstandenen Elementen der Matrix $\mathbf{M}$ sind. Die Matrix $\mathbf{M}_E$ hat nun die Form

$$\mathbf{M}_E = \begin{pmatrix} ENC_{e_{w_1}}(S_{1,1}) & ENC_{e_{w_2}}(S_{1,2}) & \cdots & ENC_{e_{w_k}}(S_{1,k}) & \cdots & ENC_{e_{w_n}}(S_{1,n}) \\ ENC_{e_{w_1}}(S_{2,1}) & ENC_{e_{w_2}}(S_{2,2}) & \cdots & ENC_{e_{w_k}}(S_{2,k}) & \cdots & ENC_{e_{w_n}}(S_{2,n}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ ENC_{e_{w_1}}(S_{j,1}) & ENC_{e_{w_2}}(S_{j,2}) & \cdots & ENC_{e_{w_k}}(S_{j,k}) & \cdots & ENC_{e_{w_n}}(S_{j,n}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ ENC_{e_{w_1}}(S_{m,1}) & ENC_{e_{w_2}}(S_{m,2}) & \cdots & ENC_{e_{w_k}}(S_{m,k}) & \cdots & ENC_{e_{w_n}}(S_{m,n}) \end{pmatrix}$$

wobei $\vec{L}_j^e := \big(ENC_{e_{w_1}}(S_{j,1}), ENC_{e_{w_2}}(S_{j,2}), \ldots, ENC_{e_{w_2}}(S_{j,k}), \ldots, ENC_{e_{w_n}}(S_{j,n})\big)$ für $1 \leq j \leq m$ bzw.
$\vec{C}_i^e := \big(ENC_{e_{w_i}}(S_{1,i}), ENC_{e_{w_i}}(S_{2,i}), \ldots, ENC_{e_{w_i}}(S_{j,i}), \ldots, ENC_{e_{wi}}(S_{m,i})\big)^T$ mit
$ENC_{e_{wi}}(\vec{C}_i) := \vec{C}_i^e$ ($1 \leq i \leq n$) wieder der j-te Zeilen- bzw. der i-te Spaltenvektor der Matrix $\mathbf{M}_E$ ist. Ein Matrix-Element $ENC_{e_{w_i}}(S_{j,i})$ ist das mit dem Public-Key e_{w_i} gemäß der Verschlüsselungstranformation $ENC(\cdot)$ chiffrierte Share des Wächters W_i zu dem Schlüssel K_j.

Proposition 5.2. Die Matrixelemente S_{ji} $(1 \leq j \leq m,\ 1 \leq i \leq n)$ seien verschlüsselt gemäß der Public-Key-Verschlüsselungsfunktion $ENC(\cdot)$.
Die Sicherheit des Schlüsselspeichers basiert auf der Sicherheit der verwendeten Public-Key-Verschlüsselungsfunktion $ENC(\cdot)$.

Beweisskizze.

1. (k-1) Wächter können zusammen mit ihren verschlüsselten Privat-Vektoren $\vec{C}_i^e$ $(1 \leq i \leq n)$ keinen der Schlüssel K_j rekonstruieren.
   Der Wächter W_i hat in diesem Fall wieder vollen Zugriff auf den Spaltenvektor $\vec{C}_i^e$, denn er kann mit seinem Privat-Key d_{e_i} gemäß $ENC_{d_{w_i}}(\cdot)$ den Vektor $\vec{C}_i^e$ dechiffrieren, so daß $ENC_{d_{w_i}}\left(ENC_{e_{wi}}(\vec{C}_i)\right) = (S_{1,i}, S_{2,i}, \ldots, S_{j,i}, \ldots, S_{m,i})^T$ für $1 \leq i \leq n$ gilt. Der Wächter W_i kann allerdings nach Proposition 5.1 keinen Schlüssel K_j $(1 \leq i \leq n)$ rekonstruieren.
   Die Komponenten des Zeilenvektors j können nur durch jeweils einen Wächter W_i mit ihrem Privat-Key d_{w_i} $i = 1, 2, \ldots, n$ dechiffriert werden, so daß der Schlüssel K_j gemäß (k, n)-Threshold-Schemas von Shamir durch k Wächtern rekonstruiert wird. $k \Leftrightarrow 1$ Wächter erhalten keinen Informationen über den Schlüssel K_j.

2. Der Schlüsselspeicher oder ein beliebiger Angreifer kann die Schlüssel K_j nur rekonstruieren, wenn sie die Verschlüsselungsfunktion $ENC(\cdot)$ gebrochen haben.

Faßt man die Ergebnisse aus Propositionen 5.1 und 5.2 zusammen, ist der Schlüsselspeicher gemäß $ENC(\cdot)$ berechnungssicher.

5.5.3 Speicherung und Zugriffsmechanismen

In der Abildung 5.3 haben wir nur den einfachen Fall betrachtet, durch den der Schlüsselspeicher konzeptionell effizient implementiert werden kann. Gemäß eines traditionellen Secret Sharing Schemas werden die Shares zu dem Schlüssel in der Verteilungsprozes vom Dealer direkt an die Wächter verteilt. Jeder Wächter behält lokal als geheime Teilinformationen seine Shares. Ein Wächter kann seine Shares beispielsweise in eine Datei speichern. Durch einfachen Zugriffsmechanismus kann der Wächter ein bestimmtes Share auslesen und es dann in der Rekonstruktionsphase ausgibt.
Anstelle der lokalen Speicherung werden alle Shares zu den eingefügten Schlüsseln im Schlüsselspeicher zentral festgehalten.

In diesem Abschnitt behandeln wir die Speicherungs- und das Managementproblem der durch den Submitter generierten Shares im Schlüsselspeicher. Danach wird zwei Konzepte für die Realisierung des *Schlüssel-Rückgewinnung.System* (RSR) am Schluß dieses Kapitels vorgestellt.

1. Fall

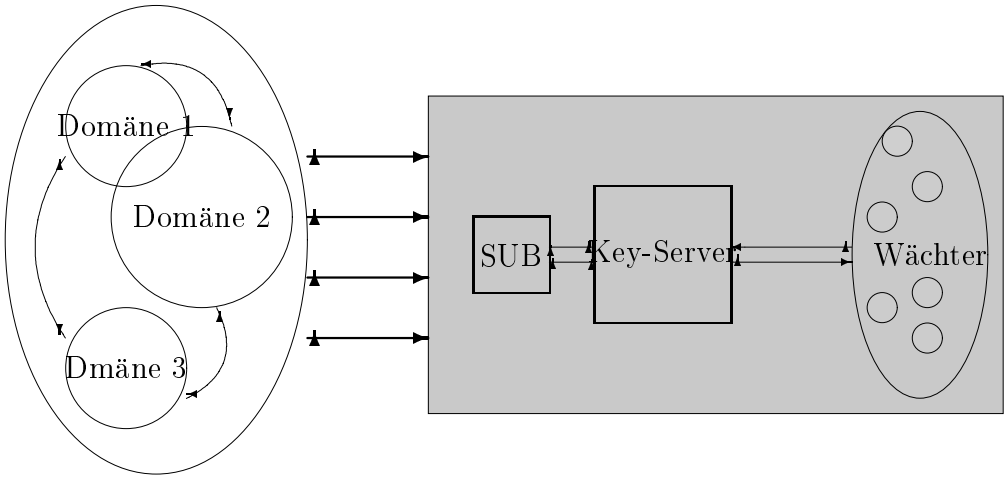

Abbildung 5.4: Schlüssel-Rückgewinnung mit mehreren Domänen.

in der Abb. 5.4 wird ein Backbone mit drei Domänen.

Formal wird jedem Benutzer U_j wird eine *Benutzeridentität* $ID(U_j)$ zugeordnet. Wenn der Benutzer U_j nur einen Schlüssel hat, dessen *Schlüsselidentität* K_j bei der Registrierung gespeichert wird, so können die eindeutige Benutzeridentität und dessen zugehörigen Schlüssel gemeinsam über den Index j indiziert werden. Die Speicherung der Shares und Zugriffsmechanismen im Schlüsselspeicher vereinfachen sich in diesem Fall,indem der Schlüssel und dessen Benutzer gemeinsam über den Index j angesprochen werden. Der Wächter W_i kann sein Share gemäß der Index-Tupel (j, i) im Schlüsselspeicher, dargestellt durch die Speichermatrix $\mathbf{M}$ die verschlüsselte Matix $\mathbf{M}_E$, zugreifen, um zusammen mit anderen Wächtern mit dem Shamir Threshold-Schema den Schlüssel mit dem Index j des Benutzers U_j zu rekonstruieren. Im folgenden wird beschrieben, daß dies ein Sonderfall ist und daß darüber hinaus die Speicherungs- und das Managementprobleme der Shares durch verschiedene *non-kryptographische Methoden* gelöst werden können.

Im allgemeinen kann ein Benutzer mehrere Schlüssel haben und er kann sie in den Schlüsselspeicher registrieren. Damit es übersichtlicher wird, führen wir einen neuen Index r ein. Wir bezeichnen $\mathcal{U} = \left\{U^{(r)} \middle| \, 1 \leq r \leq \mu\right\}$ die Menge der Benutzer, die das *Schlüssel-Rückgewinnung-System* (SRS), implementierend durch das (k, n)-Threshold-Schema von Shamir, anwenden, um ihre Schlüssel schützen zu lassen. In realistischen Computer-Netzwerken stellt die Menge $\mathcal{U}$ eine *Domäne* eines *Backbone* dar, wobei μ die maximale Anzahl der Benutzer in der Domäne $\mathcal{U}$ ist.

Wäre $\mathcal{U}$ als ein Backbone bezeichnet, das aus λ Domänen besteht, dann kann es vorkommen, daß $\bigcap_{\nu=1}^{\lambda} \mathcal{U}_\nu = X \neq \emptyset$ gilt. Diese Tatsache kommt stets in einem domänenübergreifenden Computer-Netzwerk vor, in dem ein Benutzer verschiedenen Domänen angehören kann. In Bezug auf die Speicherung und Zugriffskontrolle

spielt dies keine Rolle, da sowohl die Schlüssel als auch deren zugehörige Shares über entsprechenden Index angesprochen werden. Damit die Verwaltung der zu registrierenden Schlüssel und deren Shares einfacher wird, soll zusätzlich neben der Schlüsselidentität noch die *Schlüsselbeschreibung*, die ebenfalls in *Datenbankmethode* des zweiten Konzepts im kommenden Abschnitt vorhanden ist, seitens der Benutzer hinzugefügt werden.
Damit ist die Tatsache, daß ein Benutzer mehrere Schlüssel, die ins System registriert werden sollen, zuläsig. Man kann daher o.B.d.A. nur eine Domäne $\mathcal{U}$ untersuchen.

Wir bezeichnen ξ die maximale Anzahl der paarweise verschiedenen Schlüssel, die ein Benutzer in SRS registrieren darf. Jedem Schlüssel wird dann eine Schlüsselidentität $K_j^{(r)}$ $(1 \leq j \leq \mu \cdot \xi)$ zugeordnet, so daß über die Tupel (r, j) ein Schlüssel des Benutzers $U^{(r)}$ mit der Identität $ID(U^{(r)})$ eindeutig identiziert wird. Dies hat zur Folge, daß RSR maximal $(\mu \cdot \xi)$ Schlüssel aufnehmen kann. Gegenüber einem Anwender wird $(\mu \cdot \xi)$ als Kapazität des RSR bezeichnet, da die Shares als Systemdaten für ihn nicht sichtbar sind. Gemäß des verwendeten (k, n)-Threshold-Schema von Shamir wird jeder Schlüssel in n Shares zerlegt. Die maximale Kapazität des Schlüsselspeicher (im Bezug auf Systemsicht) ist $(n \cdot \mu \cdot \xi)$.

Für $\mu = m$ und $\xi = 1$ haben wir wieder den, wie oben erwähnt, einfachen Fall. Dabei ist K_{12}^7 beispielsweise die Identität des 12-ten Schlüssels des 7-ten Benutzers in der Domäne $\mathcal{U}$.

Da wir hauptsächlich die Speicherungs- und das Managementproblem der Shares lösen wollen, lassen wir den Index r vorläufig weg und die Schlüsselidentität $K_j^{(r)}$ vereinfacht sich zu K_j . So wird der Schlüsselspeicher ohne Berücksichtigung der Verschlüsselungsfunktion $ENC(\cdot)$ durch folgende $(\mu \cdot \xi) \times n$-Matrix dargestellt, wobei n die Anzahl der Wächter und $S_{j,i}$ das Share des Wächter W_i $(1 \leq i \leq n)$ zu dem Schlüssel K_j $(1 \leq j \leq \mu \cdot \xi)$ sind:

$$\mathbf{N} = \begin{pmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,k} & \cdots & S_{1,n} \\ S_{2,1} & S_{2,2} & \cdots & S_{2,k} & \cdots & S_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{j,1} & S_{j,2} & \cdots & S_{j,k} & \cdots & S_{j,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{(\mu\cdot\xi),1} & S_{(\mu\cdot\xi),2} & \cdots & S_{(\mu\cdot\xi),k} & \cdots & S_{(\mu\cdot\xi),n} \end{pmatrix}$$

Bezüglich der Verschlüsselungsfunktion $ENC_{e_{wi}}(\cdot)$ wird, entsprechend wie $\mathbf{M_E}$ die verschlüsselte Speichermatrix $\mathbf{N}_E$ definiert.

Im folgenden wird fest außerdem gelegt, daß der Prozess P_{KS} auf dem Schlüsselspeicher für die Speicherung der Shares zu den eingefügten Schlüsseln verantwortlich ist und auf jeden der n Wächter W_i der Prozess $P_w^{(i)}$ läuft. Entsprechend nimmt der Prozess P_{Sub} die Schlüssel eines Benutzers mit der Identität $ID(U_j)$ auf. Mit dem Algorithmus *share* des (k, n)-Threshold-Schemas generiert P_{Sub} die Shares zu dem eingefügten Schlüsseln. Unter Zuhilfenahme kryptographischer Tools sendet P_{Sub} die Shares *sicher* und *verbindlich* an den Schlüsselspeicher. Der Prozess P_{KS} speichert

die Shares mittels einer *Schreib-Operation*, so daß die Shares zeilenweise im Schlüsselspeicher vorliegen.
In der Rekonstruktionsphase muß sich jeder Wächter W_i $(1 \leq i \leq n)$ zunächst gegenüber dem Schlüsselspeicher als Wächter W_i identifizieren (Die Authentisierung werden wir später eingehen). Danach greift der Prozess $P_w^{(i)}$ auf den jeweiligen Wächter W_i den Schlüsselspeicher zu, um das zugehörigen Share $S_{j,i}$ mittels einer *Lese-Operation* auszulesen.

Nun betrachten wir das Problem: Wie kann das Shares $S_{j,i}$ ffür jedes $j \in \{1, 2, \ldots, (\mu \cdot \xi)\}$ und $i \in \{1, 2, \ldots, n\}$ durch P_{KS} und $P_w^{(i)}$ zugegriffen werden?

Bisher haben wir nur beschrieben, daß die Shares $S_{j,1}, S_{j,2}, \cdots, S_{j,k}, \cdots, S_{j,n}$ nach ihrer Erzeugung durch den Prozess P_{Sub} auf dem Submitter zum Schlüsselspeicher tranferiert werden, so daß sie dort in der Speichermatrix zeilenweise vorliegen. Allerdings haben wir nicht angegeben, mit welchen Operationen der Prozess P_{KS} auf dem Schlüsselspeicher diese Shares speichert bzw. die Wächter W_i die Shares aus dem Schlüsselspeicher für den Fall der Rekonstruktion eines Schlüssels K_j auslesen.

Basierend auf der Tatsache, daß nur der Prozess P_{KS} die von Submitter gesendeten Shares speichern kann und die Shares im Schlüsselspeicher durch die n Wächter verteilt kontrolliert wird, stellen wir drei Speicherungstechniken vor, mit denen die Zugriffsmechanismen im Schlüsselspeicher geregelt werden können. Durch diese Techniken kann der Schlüsselspeicher neben Einsatz von kryptographschen Tools vor fremden Zugriffen geschützt werden.

Matrix-Speicherung

Für die Speicherung und das Lesen der Shares werden zwei folgende Operationen $\mathsf{write}(\cdot)$ und $\mathsf{read}(\cdot)$ definiert:

- $\mathsf{write}(W_i, K_j) = S_{j,i},\ 1 \leq i \leq n$,
- $\mathsf{read}(W_i, K_j) = S_{j,i},\ 1 \leq j \leq \mu \cdot \xi$

Mit der Schreib-Operation $\mathsf{write}(W_i, K_j) = S_{j,i}$ speichert der Prozess P_{KS} im Schlüsselspeicher zeilenweise Shares $S_{j,i}$ zu dem Schlüssel mit dem Index j für den Wächter W_i, wenn der Schlüsselspeicher die Shares des nächsten Schlüssels mit dem Index j aufnimmt. Diese Operation wiederholt der Prozess P_{KS} n-mal, da der Schlüssel K_j in n Shares durch den Prozess P_{Sub} auf dem Submitter zerlegt wurde. Danach entsteht im Schlüsselspeicher der Zeilenvektor $\vec{L}_j = (S_{j,1}, \ldots, S_{j,k}, \ldots, S_{j,n})$. Wird der nächste Schlüssel eingefügt, erhöht sich der Zeilenindex der Speichermatrix um 1.

Der Prozess $P_w^{(i)}$ auf dem Wächter W_i führt die Lese-Operation $\mathsf{read}(W_i, K_j) = S_{j,i}$ aus, um das Shares $S_{j,i}$ aus dem Schlüsselspeicher auszulesen, falls Rekonstruktion aufgefordert wird.

Linearisierung der Speicher-Matrix

Bei dieser Technik wird für jede Registrierung des Schlüssels K_j der Zeilenvektor mit höherem Zeilenindex j an das Ende des Zeilenvektors mit dem niedrigen Zeilenindex

$j \Leftrightarrow 1$ angefügt, so daß eine verkettete Liste entsteht:

$$[S_{1,1}, \ldots, S_{1,n}||S_{2,1}, \ldots, S_{2,n}|| \cdots ||S_{j,1}, \ldots, S_{j,n}||S_{j+1,1}, \ldots, S_{j+1,n}|| \cdots ||S_{m,1}, \ldots, S_{m,n}],$$

wobei || die Konkatenation ist.
Die Speicheroperationen werden durch zwei Operationen $\mathsf{write}(\cdot)$ und $\mathsf{read}(\cdot)$ definiert:

- $\mathsf{write}(W_i, K_j) = S_{(j \cdot n)+i}$, wobei $(j \cdot n) + i$ die Position das Listenelement der durch zeilenweise Konkatenation entstanden ist,
- $\mathsf{read}(W_i, K_{j+1}) = S_{(j \cdot n)+i}$

Der Prozess P_{KS} führt ebenfalls die Schreib-Opreation aus, um die Shares zu speichern. Falls Rekonstruktion aufgefordert wird, führt dann jeder der Wächter W_i die Lese-Operation, um das passende Share für ihre Rekonstruktion aus dem Schlüsselspeicher auszulesen.

Permutation der Speicher-Matrix N bzw. $\mathbf{N_E}$

Die Speicherung wird zum Beispiel durchgeführt, indem die Zeilen gegen die Spalten der Matrix vertauscht werden, d.h die Matrix $\mathbf{N} = (S_{j,i})$ bzw. $\mathbf{N}_E = \left(ENC_{e_{wi}}(S_{j,i})\right)$ wird transponiert. Die beiden Operationen werden entsprechend wie folgt definiert:

- $\mathsf{write}(W_i, K_j) = S_{i,j}$,
- $\mathsf{reade}(W_i, K_j) = S_{i,j}$.

Das Transponieren der Speichermatrix $\mathbf{N} = (S_{j,i})$ bzw. bzw. $\mathbf{N_E}$ ist ein einfaches Beispiel für die Beschreibung dieser Speichertechnik. Allgemeine Methoden für die Permutation einer Matrix können aus zahlreichen Lehrbüchern der linearen Algebra entnommen werden.

Damit die Speicherung und Zugriffsmechanismen des Shares im Schlüsselspeicher kommpakt und sicherer implementiert werden kann, verwenden wir die *Datenppaselung* und einige Eigenschaften der Objektorietierung. Wir modellieren die Speichrung und das Zugriffskontroll im Schlüsselspeicher, so daß die Speichermatrix eine Instanz einer Klasse ist. Zu dieser Instanz gehören die Matrix $\mathbf{N}$ (bzw. verschlüsselte Matrix $\mathbf{N}_E$). Zu dieser Instanz gehören die Speichermatrix als Daten und die folgenden Zugriffsoperationen, nur über denen die Matrix manipuliert werden darf; und zwar ausschließlich durch den Prozess P_{KS} und die Prozesse $P_w^{(i)}$ $(1 \leq i \leq n)$:

- Der Prozess P_{KS} schreibt wahlweise mit der Operation
  1. $\mathsf{write}(S_{j,i})$ das Share $S_{j,i}$ in den Rekonstruktionsvektor $\vec{L}_j = (S_{j,1}, S_{j,2}, \ldots, S_{j,n})$ oder
  2. $\mathsf{write}(S_{j,1}, S_{j,2}, \ldots, S_{j,n})$ die ganze Zeile der Speichermatrix.

- Jeder der Prozess $P_w^{(i)}$ liest mit der Operation $\mathsf{read}(W_i, K_j)$

Damit bleibt die Speichermatrix und deren Speichertechnik in diesem Objekt *opakt* und wir können unterschiedliche Speichertechniken verwenden, die „nach außen“ nicht bekannt sind. Durch diese Methoden wird die Sicherheit der Matrix $\mathbf{N}$ (bzw. verschlüsselte Matrix $\mathbf{N}_E$) erhöht. Für den Submitter und die Wächter W_i sind schließlich nur die vordefinierten Methoden bekannt und sie bieten mehr schutz vor Datenmanipulation.
Aus diesem Ansatz bedeutet bezüglich der

- Matrix-Speicherung, daß die Shares in Form einer Matrix gespeichert werden und das Share $S_{j,i}$ dem Element in der j-ten Zeile und i-ten Zeile entspricht,
- Linearisierung der Matrix, daß die Shares linearisiert gespeichert werden und $S_{j,i}$ dem $[(j \cdot n) + i]$-te Element in der Linearisierung entspricht und
- Permutation der Matrix, daß die Shares permuntiert gespeichert werden und $S_{j,i}$ dem Element $S_{i,j}$ in der permutierten Matrix entspricht, wenn beispielsweise die Permutation dem Tranponieren einer Matrix definiert ist.

Bei der Permutation ist es zu beachten, was mit dem Zugriff geschieht, wenn ein neuer Schlüssel hinzugefügt wird. In diesem Fall muß die Matrix vor dem Schreiben rückpermutiert werden. Nachdem die neue Zeile geschrieben wurde, wird die Matrix dann permutiert.

In der Seite 107 haben wir den Index r weggelassen, damit die Speicherung der Shares und Zugriffsmechanismen im Schlüsselspeicher aus leserlichen Gründen beschrieben werden konnten. Wird nun der Index r berücksichtigt, so stellt die Speichermatrix eine einspaltige Blockmatrix mit μ Teilmatrixen dar. Jede Teilmatrix $\mathbf{B}_r$ $(1 \leq r \leq \mu)$ ist eine Speichermatrix, die entseht, wenn jeder der Benutzer $U^{(r)}$ maximal ξ Schlüssel in SRS registrieren will. Die maximale Kapazität des SRS ist offenbar $(\mu \cdot \xi)$.
Es sei $z_1, z_2, \ldots, z_\mu$ $(1 \leq z_r \leq \xi)$ die Anzahl der zu registrierenden Schlüssel des entsprechenden Benutzers $U^{(1)}, U^{(2)}, \ldots, U^{(r)}$ $(1 \leq r \leq \mu)$, so ist die Speichermatrix $\mathbf{B}_1$ für den Benutzer $U^{(1)}$ eine $(z_1 \times n)$ Matrix, $\mathbf{B}_2$ für den Benutzer $U^{(2)}$ eine $(z_2 \times n)$ Matrix und so weit und so fort, wobei n die fest vorgegebene Anzahl der Wächter W_i.
wollen alle μ Benutzer ihre Schlüssel registrieren, so sieht der Speichermatrix der gesamten Shares wie folgt aus:

$$\mathbf{N}_{gs} = \begin{pmatrix} (z_1 \times n) \Leftrightarrow Matrix \\ \ldots\ldots\ldots\ldots\ldots \\ (z_1 \times n) \Leftrightarrow Matrix \\ \ldots\ldots\ldots\ldots\ldots \\ \vdots \\ \ldots\ldots\ldots\ldots\ldots \\ (z_\mu \times n) \Leftrightarrow Matrix \end{pmatrix}$$

Die Blockmatrix $\mathbf{N}_{gs}$ ist ein $(z_1+z_2+\cdots+z_\mu)\times n$-Matrix mit $(z_1+z_2+\cdots+z_\mu) \leq \mu\cdot\xi$. Führen wir den Index r wieder ein, dann kann der Sachverhalt verallgemeinert werden. Der Benutzer $U^{(r)} \in \mathcal{U} = \left\{U^{(r)} |\ 1 \leq r \leq \mu\right\}$ hat eindeutige Benutzeridentität $ID(U^{(r)})$. Unter $ID(U^{(r)})$ wird der Schlüssel mit der Schlüsselidentität $K_j^{(r)}$ $(1 \leq j \leq \xi)$ in den Schlüsselspeicher registriert. Bezüglich des RSR gilt $1 \leq j \leq \mu \cdot \xi$. Dabei bedeutet $K_j^{(r)}$ die Identität des j-ten Schlüssels des r-ten Benutzers.
Es ist anzumerken, daß die Blockdarstellung der Matrix $\mathbf{N}_{gs}$ zustande kommt, wenn jeder Benutzer U^r hintereinander seine ξ schlüssel am Stück in SRS registriert hat. In Wirklichkeit wird jede Zeile der Matrix $\mathbf{N}_{gs}$ zu dem Zeitpunk der Registrierung eines Schlüssels gespeichert. Daher können die Zeilen der Matrix nicht gegenseitig vertauscht werden (aufgrund der Speichertechnik durch Permutation). Eine weitere Voraussetzung für die Registrierung ist: Jeder der Benutzer darauf achten muß, daß er nicht mehr als ξ Schlüssel registrieren darf, sonst wird die Systemkapazität $(\mu \cdot \xi)$ überschritten. Unter Verwendung der Datenkappselung und Opaktheit haben wir nun:
Für alle j $(1 \leq j \leq \mu \cdot \xi)$, jedes i $(1 \leq i \leq n)$ und jedes r $(1 \leq r \leq \mu)$ gilt:

- Der Prozess P_{KS} schreibt wahlweise mit der Operation
  1. $\mathsf{write}(S_{j,i}^{(r)})$ das Share $S_{j,i}^{(r)}$ in den Rekonstruktionsvektor $\vec{L}_j^{(r)} = (S_{j,1}^{(r)}, S_{j,2}^{(r)}, \ldots, S_{j,n}^{(r)})$ oder
  2. $\mathsf{write}(S_{j,1}^{(r)}, S_{j,2}^{(r)}, \ldots, S_{j,n}^{(r)})$ die ganze Zeile der Speichermatrix.
- Jeder der Prozess $P_w^{(i)}$ liest mit der Operation $\mathsf{read}(W_i, K_j^{(r)})$,

wobei $S_{j,i}^{(r)}$ das Share zu dem j-ten Schlüssel des r-ten Benutzers für den Wächter W_i. Über die Index-Tupel (r, j) wird der zu registrierende Schlüssel identiziert und über die Index-Tripel (r, j, i) weiß ein Wächter sicher, mit welchem Share er welchen Schlüssel von welchem Benutzer (zusammen mit $k \Leftrightarrow 1$ Wächtern) rekonstruiert.

Die Behandlung des Schlüsselspeichers schließen wir mit dem Hinweis ab, daß die entsprechende Speichermatrix aus Sicherheitsgründen mit der Funktion $ENC_{e_{wi}}(\cdot)$ verschlüsselt werden soll.

In der Rekonstruktionsphase muß sich jeder Wächter W_i $(1 \leq i \leq n)$ gegenüber dem Schlüsselspeicher als Wächter W_i identifizieren. Die Authentisierung kann mit dem folgenden Protokoll *authorisation-check*, das auf einem Identifikationschema (Fiat-Shamir oder Schnorr), *Password-Authentisierung* anhand des Schemas nach Lamport oder 3-*Wege-Authentisierung* nach *X.509-Standard* basiert, erfolgen [22, 30].

Bisher haben wir nur die beiden Zugriffsoperationen $\mathsf{write}(\cdot)$ und $\mathsf{read}(\cdot)$ für die Prozesse P_{KS} und $P_w^{(i)}$ $(1 \leq i \leq n)$ definiert. Damit können die Shares $S_{j,i}$ zu dem Schlüssel K_j durch P_{KS} gespeichert werden und kann jeder der Prozesse $P_w^{(i)}$ das

Element $S_{j,i}$ aus dem Schlüsselspeicher auslesen. Mit anderen Worten haben wir die Speicherung und Zugriffsmechanismen innerhalb des Schlüsselspeichers geregelt.
Da der Schlüsselspeicher bzw. die Wächter nicht wissen können, ob das Share $S_{j,i}$ zu dem Schlüssel K_j gehört. Der Index j wird nur wegen der formalen Beschreibung benutzt. Der Benutzer mit der Indentität $ID(U_7)$, kann beispielsweise 20 Schlüssel in den Schlüsselt registriert. Wie kann der Wächter W_3 das Share $S_{7_5,3}$zu dem Schlüssel K_{7_5}, d.h. der Schlüssel Nr. 5 des Benutzers Nr. 7 zugegriffen werden?.
Um diese Frage zu beantworten und darüber hinaus einen Lösungsansatz für die Verwaltung der Shares im Schlüsselspeicher angeben zu können, wird ein Datenbanksystem eingesetzt, welches wir im nächsten Abschnitt ausführlich behandeln.

5.5.4 Redundante Rekonstruktion

In Kapitel 4 wurde beschrieben, daß Jedes Secret Sharing Schema drei Haupteigenschaften hat: Teilung des Geheimnisses in Shares, Geheimhaltung des Geheimnisses durch Verteilung der Shares an verschiedene Wächter und verteilte Rückgewinnung des Geheimnisses durch autorisierte Teilmengen einer festgelegten Zugriffstruktur.

Im Kapitel 4 wurde auch gezeigt, daß durch Angriffen von „von außen“ und "Betrügen im Innen“ kann die Geheimhaltung des Geheimnisses beeinträchtigt werden. Mit der Auswahl des (k, n)-Threshold-schemas und unserer Annahme sollen die Wächter sinnvollerweise miteinander kooperieren, um eingefügten Geheimnisse zu rekonstruieren. In praktischen Anwendung, insbesondere in einem Computer-netzwerk, genügt diese Kooperation nicht, um die Geheimhaltung und redundante Rückgewinnung des Geheimnisses zu gewährleisten.

Durch kryptographische Maßnahmen, die im Kapitel 2 diskurtiert wurden, werden die Shares nach ihrer Erzeugung sicher über öffentliche Netzwerke transportiert und im Schlüsselspeicher aufbewahrt. Wenn man die Shares nicht effektiv genügend vor Angriffen schützt, dann geht die Geheimhaltung und Rückgewinnung der Schlüssel gemäß eines Secret Sharing Schemas verloren.
Damit die Schlüssel mit hoher Redundanz werden können, untersuchen wir noch die Auswirkungen von *passiven* und *aktiven* Angriffen. Durch diese Angriffe besteht die Gefahr, daß die Shares im Schlüsselspeicher und damit die zugehörigen Schlüssel aufgedeckt werden können. Die Shares werden andererseits inkonsistent, so daß die rekonstruktion nicht mehr möglich sind. Wenn die beiden Angriffsarte nicht behandelt werden, bedeutet dann eine Sicherheitsabnahme und einen Redundanzverlust für die Rekonstruierung.

Passive Angriffe bedeuteten im Rahmen dieser Arbeit, daß die Angreifer zeitlich Kenntnisse Über die Shares im Schlüsselspeicher gewinnen könnten, da die Rekonstruktion der Schlüssel im Verhälnis zu deren Registrierungen durch die Anwender seltener sind. Aktive Angriffe sind gefährlicher, denn die Angreifer versuchen nicht nur mit ihrer kryptographschen Fähigkeit die Schlüssel in Erfahrung zu bringen, sondern auch das System gegebenfalls so zu beeinflüssen, daß die Rekonstruktion gehindert wird, z.B. Manipulation der vorhandenen Shares, von Prozessen, die für

die Erzeugung bzw. Verteilung der Shares sowie Rekonstruktion der Schlüssel verantwortlich sind. Weitere aktiven Abgriffen sind zum Beispiel Beschädigung bzw. Trennung der Komponenten des Schlüssel-Rückgewinnung-Systems: Submitter, *Schlüssel-Server* (Schlüsselspeicher) und Wächter, die statt Menschen Maschinen sind. Ersetzung oder Manipulation der Shares durch aktive Angreifer führen dazu, daß die Shares nicht mehr konsistent sind. Es ist gleichgültig, ob sie lokal bei jedem der n Wächter oder zentral im Schlüsselspeicher gespeichert sind.

Dagegen sind administrative Maßnahmen dringend erforderlich

Damit die Geheimhaltung, sichere Rekonstruktion oder im Rahm der Dipmolarbeit redundante Rückgewinnung der eingefügten der kryptographischen Schlüssel noch gewährleistet werden kann, schlagen A. Herzberg, S. Jarecki, H. Krawczyk und M. Yung [14] zwei Protokolle vor, die den Schutz vor passiven bzw. aktiven Angriffen behandeln. In [14] haben sie gezeigt, daß die Lebensdauer des Geheimnisses (*long-lived-Information*)bespielsweise *CoCa-Cola* Formel, geheime Schlüssel (*master keys*) des TA (*Trusted Authority*) hoch ist. Diese geheime Information darf zu Beginn seiner Lebensdauer nur einmal mittels Secret Sharing Schemas geteilt werden darf, so daß Neugenerierung bzw. -verteilung der Shares nicht möglich ist, da man den Wert dieses dieses Schlüssels sonst ändern muß.

Die Grundidee des ersten Protokolls basiert auf der Tatsache, daß die durch Vieren infizierten Shares keine Schlüssel rekonstruieren [21] und daß passive Angreifer bzüglich der langen Lebensdauer des Schlüssels Kenntnisse von den Shares und damit dem Schlüssel gewinnen können.

Damit dieses Protokoll mit dem Schlüsselspeicher funktioniert, werden einige Protokoll-Schritte entfernt bzw. modifiziert. Dieses „abgespecktes“ Protokoll wenden wir zeitperiodisch an, um die Shares im Schlüsselspeicher vor passiven Angriffen zu schützen. Das heisst, die Werte der Shares im Schlüsselspeicher bleiben nicht mehr konstant, wie der Submitter bei der Verteilungsphase geliefert hat.

5.5.4.1 Erneuerung der Shares

Das Protokoll *refresh*

Wir bezeichnen die alten Shares im Schlüsselspeicher mit $S_{l,i}^{(r)(alt)} := S_{l,i}^{(r)}$, die zu erneuernden Shares mit $S_{l,i}^{(r)(neu)}$ und die für Generierung der alten Shares verwendeten Polynome vom Grad k mit $f_l^{(alt)}$. Dann muß nach dem (k, n)-Threshold-Schema von Shamir gelten: $S_{l,i}^{(r)(alt)} = f_l^{(alt)}(i)$ und $f_l^{(alt)}(0) = K_l^r$. Dabei bedeutet r der Benutzer- bzw. Anwender-, l der Schlüssel- und i der Wächter-Index. Statt mit dem Index j wird der Schlüssel hier mit l indiziert. Für diese Indizes gelten die Beziehungen $1 \leq r \leq \mu$, $1 \leq l \leq \xi$ und $1 \leq i \leq n$. Die Zahl μ war definiert als maximale Anzahl der Benutzer, die ihre Schlüssel in SRS registrieren lassen und ξ war definiert als maximale Anzahl der Schlüssel eines Benutzer $U^r \in \mathcal{U}$, die unter der eindeutigen Benutzer-Identität IDU^r in SRS registriert werden dürfen. Jeder registrierter

Schlüssel hat die eindeutige Schlüssel-Identitat K_l^r, die aufgrund seiner Geheimhaltung nur über der Index-Tupel (r, l) angesprochen wird. Außerdem wurde die Kapazität des SRS als $(\mu{\cdot}\xi)$ definiert. Bezüglich des verwendeten (k, n)-Threshold-Schemas von Shamir, mit dessen Algorithmus *share* jeder Schlüssel in n Shares zerlegt wird. So ist die maximale Speicherkapazität gleich $(n \cdot \mu \cdot \xi)$.

Da die Werte der Schlüssel nicht geändert werden dürfen, bleibt uns nicht übrig, die Polynome $f_l^{(alt)}$ (es gilt immer noch $f_l^{(alt)}(0) = K_l^r$) zu modifizieren. Dazu finden wir ein Polynom $\varphi \in \mathbb{Z}_p[x]$ vom Grad $k \Leftrightarrow 1$ mit der Eigenschaft $\varphi(0) = 0$ und setzen: $\forall\ i\ f_l^{(neu)}(i) := f_l^{(alt)}(i) + \varphi(i) \pmod p$, dann gilt $f_l^{(neu)}(0) = f_l^{(alt)}(0) + \varphi(0) = K_l^r$.

Wie findet man solches Polynom φ?. Nun wählt jeder Wächter W_i $(1 \leq i \leq n)$ zufällig ein Polynom $\varphi_i \in \mathbb{Z}_q[x]$ vom Grad $k \Leftrightarrow 1$ mit $\varphi_i(0) = 0$, also ohne den konstanten Term. Dann hat das gesuchte Polynom φ die Gestalt: $\varphi_(x) = \sum_{i=1}^{n} \varphi_i(x) \pmod p$. Dieses Polynom ist vom Grad $k \Leftrightarrow 1$. Das Protokoll *refresh* wird von jedem der n Wächter W_i wie folgt ausgeführt:

1. Jeder der Wächter W_i stellt eine Kommunikationsverbindung mit dem Schlüsselspeicher mittels des Protokoll *authorisation-check*$(W_i,\ KS)$ $1 \leq i \leq n$ her. Falls W_i vom KS akzeptiert wird, werden folgende Aktionen gestartet:

   $KS \Leftrightarrow\!\!\rightarrow W_i\ (1 \leq r \leq \mu),\ (1 \leq l \leq \mu \cdot \xi),\ (1 \leq i \leq n)$:

   $ENC_{e_{w_i}}(S_{l,i}^{(r)})$ und $y = SIG_{d_{ks}}\Big(H\big(ENC_{e_{w_i}}(S_{l,i}^{(r)})\big)\Big)$.

   W_i:

   - berechnet $h' = H\big(ENC_{e_{w_j}}(S_{l,i}^{(r)})\big)$.
   - $VER_{e_{ks}}(y) = ja$, falls $h' = y$
   - W_i entschlüsselt $ENC_{e_{w_i}}(S_{l,i}^{(r)})$ mit seinem Privat-Key W_{d_i} und erhält damit $S_{l,i}^{(r)}$

2. Der Wächter W_i wählt zufällig $k \Leftrightarrow 1$ Koeffizienten $a_{i,1}, a_{i,2}, \ldots, a_{i,k-1}$ in $\mathbb{Z}_p$. Dann hat das Polynom φ_i die Gestalt $\varphi_i(x) = a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,k-1}x^{k-1}$ und es gilt $\varphi_i(0) = 0$ für alle i.

3. Jeder der Wächter W_i berechnet $u_{ij} = \varphi_i(j) \pmod p$ für alle $j \neq i$. Sicher und verbindlich sendet der Wächter W_i anschließend u_{ij} jeweils an die anderen Wächter W_j $(j \neq i)$ unter Verwendung von $ENC_{e_{w_j}}(\cdot)$ und $SIG_{d_{w_i}}(\cdot)$, wobei e_{w_j} der authentische Public-Key von jedem W_j und d_{w_i} der Privat-Key von W_i mit dem folgenden Schema:

   $W_i \Leftrightarrow\!\!\rightarrow W_j$:

   $ENC_{e_{w_j}}(u_{ij})$ und $y = SIG_{d_{w_i}}(H(u_{ij}))$.

   W_j:

- Der Wächter W_j entschlüsselt mit deinem Privat-Key d_{w_j} und erhält damit u_{ij}.
- Gemäß der öffentlichen Hashfunktion H berechnet W_j den Wert $h' = H(u_{ij})$.
- Schließlich überprüft W_j mit dem öffentlichen Verifikationsalgorithmus $\mathsf{Sig}(\cdot)$: $\mathsf{Ver}_{d_{wi}}(y) = ja$, falls $h' = H(u_{ij})$.

4. Genau wie die Überprüfung der u_{ij} durch W_j überprüft der Wächter W_i die u_{ji} (Indexvertauschung), die der Wächter W_i von anderen Wächtern W_j für alle $j \in \{1, 2, \ldots, n\}$ veröffentlicht wurden, und berechnet seine neue Shares mit $S_{li}^{(r)(neu)} = S_{l,i}^{(r)(alt)} + u_{1i} + u_{2i} + \cdots + u_{ni} \pmod p$. Diese Berechnung ist möglich, da $S_{l,i}^{(r)} = S_{li}^{(r)(alt)}$.

5. Außer $S_{l,i}^{(r)(neu)}$ löscht W_i alle verbleibenden Parameter: $S_{l,i}^{(alt)}$ und u_{ji}.

6. W_i senden nun sicher und verbindlich sein neues Share $S_{l,i}^{(neu)}$ an den Schlüsselspeicher. Dort werden alle alten Werte $S_{l,i}^{(alt)}$ von dem Proress P_{KS} gelöscht.

Nach dem Lemma 5.1 sind die Shares $S_{l,i}^{(r)(neu)}$ konsistent. Es ist anzumerken, daß die Shares nach der Erneuerungsphase inkonsistent werden. Dies kann nur der Fall sein, wenn die W_i während der Berechnug ihrer neuen Shares $S_{li}^{(neu)}$ angegriffen worden sind, bevor sie zum Schlüsselspeicher gelangen.
Unter der Voraussetzung, daß jeder der Wächter W_i mit der Berechnung ihrer neuen Shares $S_{li}^{(neu)}$, dann können wir durch folgende Proposition zeigen, daß k neue Shares den Schlüssel K_l^r sicher rekonstruieren.

Lemma 5.1. Mit n Shares $S_{li}^{(neu)}$ können k Wächter gemäß des (k, n)-Threshold-Schema von Shamir dieselbe Schüssel K_l^r rekonstruieren.

Beweis. Es genügt zu zeigen, daß k im Protokoll *refresh* durch n Wächter neu erzeugte Shares $S_{l,1}^{(r)(neu)}, S_{l,2}^{(r)(neu)}, \cdots, S_{l,k}^{(r)(neu)}$, für jedes r, l mit $1 \leq r \leq \xi$ und $1 \leq l \leq \mu \cdot \xi$ den Schlüssel mit der Identität K_l^r rekonstruiert.

Anstelle der zufällig gewählten $x_{j,1}, x_{j,2}, \ldots, x_{j,k}, \ldots, x_{j,n}$ Stützstellen aus $\mathbb{Z}_p$ benutzen wir hier den Wächter-Index i. Dann gilt für die alten Shares:
$S_{l,i}^{(r)(alt)} = f_l^{(alt)}(i)$. Vor der Share-Erneuerung rekonstruieren k Shares den Schlüssel K_l^r, d.h. $f_l^{(alt)}(0) = K_l^r$. Da der Schlüssel auch anhand der Lagrange'schen Interpolationsformel mit den k alten Shares rekonstruiert werden, gilt weiter:

$$f_l^{(alt)}(x) = \sum_{s=1}^{k} \Big(\prod_{1 \leq t \leq k, t \neq s} \frac{x \Leftrightarrow i_t}{t_s \Leftrightarrow i_t} \Big) \cdot f_l^{alt}(i_s)$$

Für $x = 0$ sind $L_s = \prod_{1 \leq t \leq k, t \neq s} \frac{i_t}{t_s \Leftrightarrow i_t}$ Lagrange'sche Koeffizienten. Also muß gelten

$$K_l^r = f_l^{(neu)}(0) = \sum_{s=1}^{k} L_s \cdot f_l^{(alt)}(i_s).$$

Durch Umindizieren kann die obige Formel so schrieben werden:

$$K_l^r = \sum_{i=1}^{k} L_i \cdot f_l^{(alt)}(i).$$

Da die Polynome $\varphi_j(\cdot)$ vom Grad $(k \Leftrightarrow 1)$ sind und haben keinen konstanten Term, so muß für jedes Polynom φ_j gelten: $\sum_{i=1}^{k} L_i \cdot \varphi_j(i) = \varphi_j(0) = 0$. Mit anderen Worten: k Werte $\varphi_j(i)$ $1 \leq j \leq n$ interpolieren $\varphi_j(0)$. Dieses Ergebnis brauchen wir für den Beweis. Für die neuen Shares $S_{l,i}^{(r)(neu)}$ gilt:

$$\begin{aligned}
\sum_{i=1}^{k} L_i \cdot S_{l,i}^{(r)(neu)} &= \sum_{i=1}^{k} L_i \left(S_{l,i}^{(r)(alt)} + u_{1i} + u_{2i} + \cdots + u_{ni} \right) \\
&= \sum_{i=1}^{k} L_i \left(S_{l,}^{(r)(alt)} + \varphi_1(i) + \varphi_2(i) + \cdots + \varphi_n(i) \right) \\
&= \sum_{i=1}^{k} L_i \left(S_{l,}^{(r)(alt)} + \sum_{j=1}^{n} \varphi_j(i) \right) \\
&= \sum_{i=1}^{k} L_i S_{l,i}^{(r)(alt)} + \sum_{j=1}^{n} \sum_{i=1}^{k} L_i \cdot \varphi_j(i) \\
&= K_l^r + \sum_{j=1}^{n} \varphi_j(0) = K_l^r.
\end{aligned}$$

Da der Beweis für jedes $r \in [1, \mu]$ und jedes $l \in [1, (\mu \cdot \xi)]$ gültig ist, können die gesamten $(n \cdot \mu \cdot \xi)$ Shares auf dieser Weise erneuert werden.
Setzen wir $(neu) := t$ und $(alt) := t \Leftrightarrow 1$, wobei t die Zeitperiode, nach der die Gesamtheit der Shares erneuert werden sollen, so kann die Share-Erneuerung automatisch beispielsweise jede Woche durchgeführt werden.

5.5.5 Authentisierung im SRS

Das Protokoll *authorisation-check*

Anhand der Public-Key-Verschlüsselungstranformation $ENC(\cdot)$ und des Signature-Schemas $SIG(\cdot)$ werden die Geheimhaltung, Verbindlichkeit und Integrität im SRS gesichert. Die Sicherheit des Schlüsselspeichers, in dem die Shares aller registrierten Schlüssel gespeichert sind, wird vor Systemkommponenten (Submitter, Schlüsselspeicher) und unbefugten Personen durch eingebetten kryptographischen Tools und insbesondere das eingesetzte (k, n)-Threshold-Schema geschützt. Durch eingeführte Speichertechniken erhöht sich die Zugriffssicherheit des Schlüsselspeichers. Allerdings

können wir nicht verhindern, daß sich eine beliebige Person ins SRS meldet und seine Schlüssel registrieren.

Um das Konzept zu vervollständigen, stellen wir ein Protokoll vor, *authorisation-check*, mit dem die Benutzer/Anwender des SRS identifiziert werden. In das Protokoll wird wahlweise ein der Identifikationsschemata für die Authentisierung eingebettet (Unterabschnitt 2.4.4.3, Seite 42).

Mit diesem Protokoll kann sich jeder der Kommunikationsteilnehmer gegenüber einem anderen identifizieren, um eine Kommunikationsverbindung herzustellen. Abhängig von der Systeminfrastruktur soll geeignetes Identifikationschema in dem Protokoll verwenden. So wird beispielsweise das Authentisierungsschema durch Ein-Weg-Password von L. Lamport [17] ausgewählt, wenn sich die Benutzer am *Sicherheits-Server* melden, um ihre Schlüssel registrieren zu lassen. Der Grund ist: Die Anzahl der Registrierung kann sehr hoch sein. Statt [8, 9] kann das Schema von Schnorr [20] bzw. T. Okamoto [19] verwendet werden, da der Kommunikationsaufwand zwischen dem Submitter und dem *Schlüssel-Server* als Sclüsselspeicher groß ist. Da die Rekonstruktion der Schlüssel aufgrund unserer Anforderung sicher und zuläsig erfolgen soll, soll das Identikationsschema von Fiat und Shamir [8] in das Protokoll *authorisation-check* eingebettet und von autorisierten Wächtern gegenüber dem Schlüssel-Server (Schlüsselspeicher) durch geführt werden.

Das Protokoll wird mit zwei Parametern A und B ausgeführt, wobei $A := W_i$ und $B := P_{KS}$ in dem folgenden Authetisierungsschema zwischen k Wächtern und dem Prozess P_{KS} auf dem Schlüssel-Server gesetzt sind. Für jedes $r \in \{1, 2, \ldots, \mu\}$ $j \in \{1, 2, \ldots, \mu \cdot \xi\}$ und $i \in \{1, 2, \ldots, n\}$ sieht die Aktion des Protokolls *authorisation-check*$(W_i, Schlüsselspeicher)$ wie folgt aus:

1. Der Wächter W_i will das Share $S^r_{j,i}$ zugreifen. Er sendet eine Anfrage an den Schlüsselspeicher.
2. Mittels eines geeigneten Identifikationsschemas überprüft P_{KS}:
   (a) Falls W_i autorisiert ist, an die Rekonstruktion des Schlüssels K_j zu beteiligen, dann erhält der Wächter W_i als Antwort „ACCEPT“,
   (b) sonst „REFUSE“.
3. Ist die Authentisierungsvorgang erfolgreich abgeschlossen, kann der Wächter W_i mit einer geeigneten Lese-operation sein Share zugreifen.
4. Der Wächter W_i dechiffriert das verschlüsselte Share mit seinem Privat-Key d_{w_i} und gibt es für die Schlüsselrekonstruktion aus.

Die Abbildung 5.5 beschreibt alle Authentisierungsvorgänge, die von der Eingabe der Geheimnisse durch Benutzer bis zur Rekonstruktionsphase durch die k Wächter erstreckt.

Die Begriff *Sicherheits-Server* und *Schlüssel-Server* wurden verwendet, denn wir beschreiben im folgenden Abschnitt **das erste Gesamtkonzept** des Schlüssel-Rückgewinnung-Systems SRS.

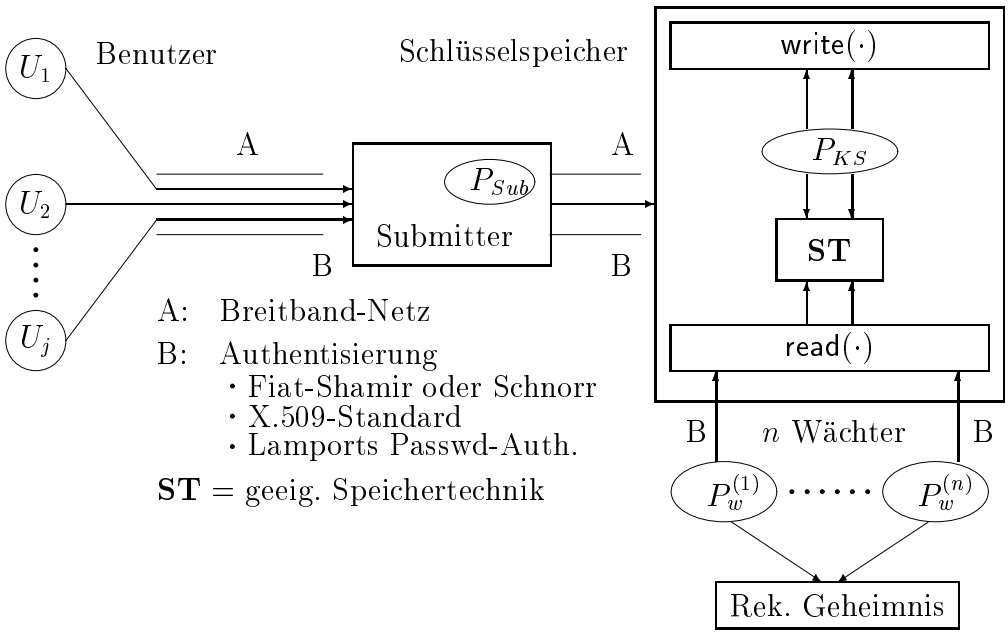

Abbildung 5.5: Authentisierung und Zugriffsmechanismen

5.5.6 Redundanz-Verlust verhindern

Jedes (k, n)-Threshold-Schema erlaubt, daß höchstens $(n \Leftrightarrow k)$ Shares korrupt werden dürfen. Wenn die Shares im Sclüsselspeicher bzw. die Wächter beschädigt werden, wird die Rekonstruktion unmöglich wird. Durch eine suksessive Vernichtung der Shares nimmt die Redundanz der Recovery ab. Daher ist es notwendig, diese Beschädigung zu verhindern.

Mit dem Protokoll *Share-recover* können die beschädigten Shares rekonstruiert werden, so daß die Rekonstruktion des Schlüssels nicht gefährdet werden kann. Das Protokoll kann modofiziert werden, um beschädigte Shares zurückzugewinen [5].

5.6 Das erste Konzept des Relaisierens eines SRS

Die Registrierung und Rückgewinnung der Schlüssel mittels des SRS besteht kozeptionell eigentlich aus vier Phasen:

Initialisierung

1. Als Sicherheitsparameter wird eine Primzahl p festgelegt. Alle Rechenoperationen finden im Restklassenkörper $\mathbb{Z}_p$ statt.

2. Der Sicherheits-Server legt ein Identifikationsschema aus [8, 9, 20, 19, 17] zur Authentisierung fest. Als Beispiel wird das Einweg-Password-Schema nach L. Lamport [17].

3. Die Mazimale Anzahl der Benutzer μ sowie der zu registrierenden Schlüssel ξ für alle Benutzer werden vom System festgesetzt.

4. Das (k, n)-Threshold-schema mit entsprechender Parametrisierung (Auswahl von k und n) wird fest gelegt, so daß $W = \{W_1, \ldots, W_n\}$ die Wächtermenge ist und k Wächter ausreichen, um die im SRS registrierten Schlüssel zu rekonstruieren. Der Algorithmus *share* auf dem Submitter wird durch CA signiert. Dies kann durch den Sicherheits-Server übernommen.

5. Eine entsprechende Speichertechnik wird festgelegt, wobei die Schreib-Operation write$(\cdot)$ nur dem Prozess P_{KS} auf dem Schlüsselserver und die Lese-Operation read$(\cdot)$ nur jedem der Prozesse $P_w^{(i)}$ bekannt sind.
   Der Algorithmus *recover* auf jedem Wächter W_i wird ebenfalls durch CA signiert.

6. Es gibt ein Zertifikat-Verzeichnis, vom dem jede Komponente, falls erforderlich ist, Zertifikat der anderen holen kann. Jedes Zertifikat wird ebenfalls durch von CA signiert.

7. Benutzer im Lan, die ihre Schlüssel ins SRS registrieren wollen, müssen eine Zugangsberechtigung haben. Die Zugangsberechtigung setzt sich aus:

   (a) Benutzername U^r, dem eine eindeutige Benutzeridentität $ID(U^r)$ vom System zugeordnet wird (der Index r entspricht dieser Benutzeridentität).$ID(U^r)$ ist Systemdaten und bleit dem Benutzer unsichtbar.

   (b) Standard-Password.

   Die Benutzerdaten sind Benutzername und Password. Das Password soll aus Sicherheitsgründen vom Benutzer beliebig geändert werden, während die Daten Benutzername vom Benutzer nicht geändert werden kann.

Generierung der Shares

Registriert der Benutzer U^r ein Schlüssel, dann werden folgende Schritte durchgeführt:

- Der Benutzer U^r gibt seinen benutzername und sein Password ein (Authentisierung findet mit dem Protokoll $authorisation \Leftrightarrow check(U^r, Submitter)$ statt. Die Kommunikationsverbindung zwischen U^r und Submitter wird hergestellt.

- Der Benutzer gibt seinen Schlüssel ein. Anhand der Login-Daten r wird ein Schlüsselidentität (r, j) vom System erzeugt und gespeichert (die Bezeichnung K_j^r ist mit $(r, j) gemeint$). Der Submitter wählt bezüglich j zufällig ein Polynom $f_j(x) \in \mathbb{Z}_p[x]$ vom Grad $k \Leftrightarrow 1$ und berechnet die entscprechenden Shares

$S^r_{j,1}, S^r_{j,2}, \cdots, S^r_{j,k}, \cdots, S^r_{j,n}$. Schließlich wendet er entscprechend $ENC(\cdot)$ und $SIG(\cdot)$ auf die Shares an und sendet die signierten, verschlüsselte Shares an den Schlüsselspeicher.

Speicherung der Shares

- Der Schlüsselspeicher speichert die Shares nach Entschlüsselung und Verifikation zeilenweise mit der vordefinierten Schreib-Operation.
  Die Shares ist nun die neue Zeile der verschlüsselten Speichermatrix $\mathbf{N}_E$.

Rekonstruktion

Soll der Schlüssel mit der Schlüsselidentität (r, j) rekonstruiert, wird eine autorisierte Teilmenge der Wächter vom System entschieden. Die k Wächter führen die folgenden Schritte durch:

- jeder der k Wächter führen zusammen mit dem Schlüsselspeicher das Protokol $authorisation(W_i, Schlüsselspeicher)$ $i \in \{1, 2, \ldots, k\}$ vollständig aus.
- Danach wird die Kommunikationsverbindung hergestellt. Jeder der k Wächter greifen anhand der Information (r, j, i) und der vordefinierten Lese-Operation $\mathsf{read}(\cdot)$, um das Share $S^r_{j,i}$ zu holen.
- Mit den Shares $S^r_{j,1}, S^r_{j,2}, \cdots, S^r_{j,k}$ rekonstruieren die k Wächter gemeinsam mit dem Protokoll *recover* des (k, n)-Threshold-Schema den Schlüssel mit der Schlüsselidentität (r,j) rekonstruiert.
- Die Wächter übergeben den Schlüssel der vertrauenwürdiggen *Ausgabestelle*
- Nach dem Prinzip von Secret Sharing sollen die Shares nur für Pro Rekonstruktion benutzt werden. Die Shares werden gelöscht und damit auch die Schlüselidentität j, jedoch nicht die Benutzeridentität r.
- Je nach der *Sicherheitspolitik nach der Rekonstruktion* und abhängig von dem verwendeten Secret Sharing Schema wird entschieden, was mit dem rekonstruierten Schlüssel geschieht.

5.7 Das zweite Konzept für die Realisierung eines SRS

Basierend auf der Ergebnisse des ersten Konzepts wird hier einen anderen Ansatz vorgestellt. Aus diesem Ansatz wird die Verwaltung der Shares durch ein Datenbanksystem eingesetzt. Während zusätzliche Sicherheit der Shares im Schlüsselspeicher

durch die Einführung der Zugriffsoperationen gesichert wird, nutzen wir hier die Sicherheitseigenschaft einer Daten bank aus, um die Speicherung und Zugriffssregelung der Shares zu organisieren. Bezüglich des Schlüsselspeichers kann ferner die verteilte Datenbank eingesetzt werden, um die Rückgewinnungseigenschaft zu erhöhen.

Um die hier verwendeten Begriffe terminologisch eindeutig festzulegen, bezeichnen wir die zu registrierenden Schüssel als Geheimnisse, da der Begriff *Primärschlüssel* hier verwendet wird. Weiter fassen wir die drei Komponenten Submitter, Schlüsselspeicher und die Wächter W_i zu einem System, bezeichnet als *Schlüssel-Rückgewinnung-System* (SRS), zusammen. Die Benutzer, die ihre Geheimnisse ins SRS wollen, haben aus Sicherheitsgründen keine direkten Verbindungen zu Submitter, Schlüsselspeicher und Wächtern. Die Benutzer sind über ein Breitband-Netz mit dem Submitter verbunden, so daß sie beispielsweise über einen WEB-Browser ihre Geheimnisse in den Submitter eingeben können. Bei jeder Registrierung findet die Authentisierung statt, welche auf Identifikationsschema (Fiat-Shamir oder Schnorr) oder auf dem X.509-Standard basiert. Einfachheit halber wählen wir die *Password-Authentisierung* anhand des Schemas von Lamport, um unseren Lösungsansatz zu demonstrieren. Bei dieser Methode hat jeder Benutzer drei Komponenten für seine Identifizierung. Das sind: *BenutzerIdentität, Dame* und *Password*. Das Geheimnis sei ein Element aus dem Restklassenkörper modulo einer Primzahl p, in Zeichen $\mathbb{Z}_p$.

Nun realisieren wir die Speicherung der Shares und Zugriffsmechanismen auf dem Schlüsselspeicher mit Hilfe eines Datenbanksystems. Dafür müssen die zum SRS anfallenden Daten und deren Beziehungen untereinander *analysiert* und entsprechend *modelliert* werden.

5.7.1 Datenmodellierung im Recovery-System

Zu den daten des Schlüssel-Rückgewinnung-Systems gehören die charakteristischen Benutzerdaten, die Daten der im System aufzubewahrenden Geheimnisse (jedoch nicht die Geheimnisse selbst) und schließlich die Daten zur Rekonstruktion der eingefügten Geheimnisse. Gemäß der Eigenschaft der Datenautonomie in der Datenbank lassen sich die drei Datengruppen durch drei unabhängigen Relationen bzw. Tabellen *User, Secret* und *Shadows* modellieren.

Die Relation *User* beschreiben die Benutzer, die ihre Geheimnisse im RSR aufbewahren lassen. Die Relation enthält drei Attribute: *UserIdentity*, *UserName* und *Password*.

Das Attribut *UserIdentity* ist der *Primärschlüssel* zur Identisierung der Benutzer und wird beim Anlegen eines neuen Benutzers vom System generiert. Die beiden Attribute *UserName* und *Password* können vom Benutzer ausgewählt werden und werden für die Authentisierung bzw. eine Berechtigung zu dem SRS benötigt. Dies ist vergleichbar mit der Authentisierung eines Benutzers in einem Multiuser-Plattform, zum Beispiel in einem UNIX-System mittels der Datei *Yellow Page*. Zu diesen drei Attributen können noch weitere Felder dazu definiert werden, die den Benutzer mehr systemcha-

rakteristisch beschreiben können. Damit entspricht jede Zeile in dieser Relation die Definition eines Benutzers, der durch seine Attribute beschrieben, identifiziert bzw. authentisiert wird.

Die Relation *Secret* speichert die im System eingefügten Geheimnisse. Jedes Geheimnis wird intern einen eignen Schlüssel *SecretIdentity* identifiziert, da wir keine Geheimnisse direkt in der Datenbank speichern dürfen. Für die Zurückgewinnung der Geheimnisse sind die k von n Wächtern verantwortlich. Damit jeder Benutzer die Möglichkeit hat, jedes seiner Geheimnisse mehr zu beschreiben, fügen wir noch das Attribut *SecretDecription* hinzu.

Die Speichermatrix $\mathbf{M}_E$ (aus leserlichen Gründen wird die Verschlüsselungsfunktion $ENC_{e_{wi}}(\cdot)$), durch welche die Speicherung der Shares dargestellt ist, wird anhand der Ralation *Shadows* beschrieben. Dabei entspricht die Anzahl der Relationspalten der Anzahl der n Wächter, die in SRS vordefiniert sind. Die Speichermatrix $\mathbf{M}_E$ läßt sich $1:1$ auf die Relation abbilden. Jedem Zeilenvektor von $\mathbf{M}_E$ wird eine Relationszeile zugeordnet. Um die einzelnen Shares identifizieren zu können, ist wie bei jeder obigen Relation ein Schlüsselattribut als Primärschlüssel notwendig.

5.7.2 Modellieren der Beziehungen

Jede der gefundenen Relationen stellt seine eigene Welt dar und die Relationen können nur durch die noch zu modellierenden Beziehungen die gesamte Welt des SRS wiedergeben. In der Realität besitzt jeder Anwender dieses Systems nicht nur ein Geheimnis, sondern er kann mehrere Geheimnisse haben. Damit entsteht ein sog. $1:n$ *Beziehung* zwischen den Relation *User* und *Secret*. Dagegen ist jedes Geheimnis nach unserem SRS mit genau einer Zeile der Speichermatrix $\mathbf{M_E}$ bzw. der Relation *Shadows* assoziert und entsteht somit eine $1:1$ *Beziehung*. Diese Tatsache kann im *ER-Modell* (*Entity Relationship Model*) modelliert werden. Das Beziehungsmodell wird in der Abbildung 5.6 dargestellt. Wir nennen die $1:n$ *Beziehung* von der Rela-

Abbildung 5.6: Beziehungsmodell

tion *User* zu der Relation *secret* die *Having-Secrets* und die $1:1$ *Beziehung* zwischen beiden Relationen *secret* und *Shadows* die *Having-a-Share Beziehung*. Die Zahl 1 auf der Seite der Relation *User* und n auf der Seite der Relation *Secret* in der Abbildung 5.6 drücken zusammen aus, daß eine Zeile in der Relation *User* mit mehreren Zeilen in der *Secret* in Beziehung stehen kann und dies entspricht unserer Realität. Wird ein Datenbanksystem zur Realisierung des SRS ausgewählt, so kann die $1:n$ Beziehung

Having-Secrets durch eine eigene Tabelle beschrieben werden, die nur zwei Attribute besitzt. Das erste Attribut ist ein numerischer Schlüssel (*UserIdentity*), der die Identität eines Benutzers in der Tabelle *User* im Schlüssel-Recovery-System darstellt und das zweite ist ebenfalls ein numerischer Schlüssel (*SecretIdentity*) zur Identifizierung der in SRS eingegebene Geheimnisse, da keine Geheimnisse dort aufbewahrt werden dürfen. Das Schlüsselpaar (*UserIdentity*, *SecretIdentity*) ist dadurch der Primärschlüssel zur Verknüpfung der Tabelle *User* mit der Tabelle *Secret*. Die zweite Beziehung *Having-a-Share* läßt sich einfacher realisieren als die *Having-Secrets*. Hier braucht man die Identität *SecretIdentity* als Primärschlüssel in die Relation *Shadows* aufzunehmen. Auf gleicher Weise können wir den Primärschlüssel *UserIdentity* in die Relation *Secret* aufnehemn und die $1 : n$ Beziehung *Having-Secrets* läßt durch eine geeignete Selektion auf dieser Relation realisieren.

5.7.3 Datenbeispiel

Im folgenden Datenbeispiel sollen die Daten- und Beziehungsmodellierung verdeutlicht werden. In der Tabelle 5.1 wird jede Relation mit einer willkürlichen Daten gefüllten Tabelle präsentiert.

UserIdentity	**UserName**	**Password**
11	Otto	VerschlüsseltesOttoPassword
22	Hugo	VerschlüsseltesHugoPassword
33	Motto	VerschlüsseltesMottoPassword
⋮	⋮	⋮

Tabelle 5.1: Die Relation *user*

Die Relation *User* enthält drei Benutzer. Das „verschlüsselte Password“ in der Spalten **Password** ist das Kennwort für den Authentisierungsvorgang jedes Benutzers. Mit diesem Kennwort kann der Benutzer Otto beispielsweise dem Submitter gegenüber als Otto identifizieren - andere Möglichkeit wäre 3-Wege-Authentisierung nach X.509-Standard. „Verschlüsseltes Password“ kann durch Verwendung von Ein-Weg-Hashfunktion, zum Beispiel das Schema nach Lamport, realisiert werden. Nach der Authentisierung kann der Benutzer Otto mit der Registrierung seiner Geheimnisse beginnen.

Die Relation *Secret* ist eine Sammlung aller bisher in Schlüssel-Recovery-System eingefügten Geheimnisse und hat beispielsweise 8 Einträge. Durch geeignete Selektion z.B. SQL-Anweisungen, welche die $1 : n$ Beziehung *Having-Secrets* beinhaltet, können wir feststellen, daß der Benutzer Otto mit der *Identitätsschlüssel* 11 bereits drei Geheimnisse in den Schlüsselspeicher eingefügt, nämlich:

(1, *Decription* 1, 11), (3, *Decription* 3, 11) und (7, *Decription* 7, 11)

Die Relation *Secret* wird durch drei Attribute *SecretIdentity, Description* und *UserIdentity* in der Tabelle 5.2 beschrieben.

SecretIdentity	Description	UserIdentity
1	Description 1	11
2	Description 2	33
3	Description 3	11
4	Description 4	22
5	Description 5	33
6	Description 6	22
7	Description 7	11
8	Description 8	33
⋮	⋮	⋮

Tabelle 5.2: Die Relation *Secret*

SecretIdentity	W_1	W_2	$\cdots$	W_n
1	$S_{1,1}$	$S_{1,2}$	$\cdots$	$S_{1,n}$
2	$S_{2,1}$	$S_{2,2}$	$\cdots$	$S_{2,n}$
3	$S_{3,1}$	$S_{3,2}$	$\cdots$	$S_{3,n}$
4	$S_{4,1}$	$S_{4,2}$	$\cdots$	$S_{4,n}$
5	$S_{5,1}$	$S_{5,2}$	$\cdots$	$S_{5,n}$
6	$S_{6,1}$	$S_{6,2}$	$\cdots$	$S_{6,n}$
7	$S_{7,1}$	$S_{7,2}$	$\cdots$	$S_{7,n}$
8	$S_{8,1}$	$S_{8,2}$	$\cdots$	$S_{8,n}$
⋮	⋮	⋮	⋮	⋮

Tabelle 5.3: Die Relation *Shadows*

Die Tabelle 5.3 beschreibt die Relation *Shadows*. In dieser Tabelle ist n die Anzahl der Wächter. Im gegensatz zu der Relation *Secret* werden alle Shares, welche zur Rekonstruktion der eingefügten Geheimnisse von n Wächtern benötigt werden, hier gespeichert. Wird das Geheimnis (7, *Description*7 , 11) rekonstruiert, muß jeder der k Wächter W_i sein Share $S_{j,i}$ aus der Relation *Shadows* über einem vordefinierten *Dienst* auslesen, welches entsprechend in der Zeile 7 der Tabelle *Shadows* zu finden ist (Tabelle 5.3). Diese Shares geben die k Wächter weiter an den Algorithmus *recover* des (k, n)-Threshold-Schemas von Shamir, um das zugehörige Geheimnis zu rekonstruieren.

5.7.4 Dienste im SRS

Ein Benutzer, der seine Geheimnisse einfügen will, muß dem Submitter bekannt sein. In dem ersten Konzept muüssen die Benutzer mit dem Submitter das Protokoll *authorisation-check* vollständig durchführen, sonst können sie keine Schlüssel ins SRS eingeben. Diesen Schritt wird durch einen Dienst der Datenbank ersetzt, denn jede Datenbank muß in der Lage sein, Zugangsberechtigung die Benutzer erstellen können.

Hierbei stellen wir einige notwendigen Hauptdienste aus der Benuzer- und Systemsicht vor. Für die Benutzer sind folgende Dienste von Bedeutung:

Benutzer-Dienste

- Durch Eingaben von *UserName* und *Password* können sich die Benutzer zuerst am SRS anmelden.
- Falls die Authentisierung erfolgreich abgeschlossen ist, kann die Benutzer ihre neue Geheimnisse registrieren lassen. Als Eingabe sind Geheimnisse und deren Beschreibung.
- Falls ein Geheimnis rekonstruiert werden soll, kann es über seine Identität zückgewonnen werden. Als Eingabe für diesen Dienst ist die *SecretIdentity*.

System-Dienste

1. Einen neuen Benutzer ins SRS aufnehmen.
2. Anmeldung der Benutzer ins SRS kontrollieren.
3. Ein neues Geheimnis aufbewahren, d.h. die Shares zu dem Geheimnis in der Datenbank speichern.
4. Ein Geheimnis nach Anforderung über seine Identität rekonstruieren.

Insgesamt werden die Schlüsselregistrierung und -rückgewinnung hauptsächlich in drei Phasen durchgeführt:

Erstellung der Zugangsberechtigung der benutzer

In dieser Phase werden die Authentisierungsinformationen, die für Anmeldung der Benutzer vom System erstellt und gespeichert. Diese Schritte werden durch Sicherheitsadministrator durchgeführt:

- Das System generiert eine *UserIdentity* für den neuen Benutzer.
- Der Benutzer erhält einen *UserName* für seine Anmeldung
- Das System vergibt ein *Standard-Password*, das später durch den Benutzer geändert werden und sich am System anmelden kann.
- Abschließend wird eine Zeile:

  (UserIdentity, UserName, verschlüsseltes Password)

  in die Relation *User* aufgenommen.

Registrierung eines neuen Geheimnisses

In dieser Phase Der zweite Dienst ermöglicht einem dem SRS bekannten Benutzer, daß er seine Geheimnisse in den Submitter eingeben kann. Die folgenden Schritte vervollständigen den Einfügungsvorgang eines Benutzers:

1. Der Benutzer loggt sich erfolgreich mit seinen Identifikationsdaten ins SRS ein. (sonst ist er nicht berechtigt, irgendein Geheimnis einzugeben).
2. Der Benutzer gibt sein Geheimnis und dessen Beschreibung ein.
3. Das SRS generiert eine Identität für dieses Geheimnis.
4. Die entstandene Zeile *(SecretIdentity, Decription, UserIdentity)* wird in die Relation *Secret* eingefügt.
5. Das System generiert die Shares *(SecretIdentity,* $S_{j,1}, S_{j,2}, \ldots, S_{j,k}, \ldots, S_{j,n}$*)* wird in die Relation *Shadows* eingefügt.

Löschung eines Eintrags Nachdem ein Geheimnis rekonstruiert worden ist, soll das Geheimnis mit einer Gültigkeitsdauer versehen, damit kein Mißbrauch möoglich ist. Da unsere Konzeption auf der relationellen Datenbank basiert, werden die zugehörigen Einträge anderer Relationen gelöscht. Als Eingabe für die Löschung ist die *SecretID*. Das folgende Beispiel veranschaulicht die Löschung eines Eintrags nach einer Rekonstruktion.
Angenommen, das Geheimnis mit dem *SecretID*= 3 in der Relation *Secret* soll gelöscht werden, dann wird die $UserID = 11$ und $SecretDecrytion = Document$ 3 mit gelöscht, jedoch nicht die $UserID = 11$ in der Relation $UserName$. Es hat zur Folge, daß die dritte Zeile $\left(3, (S_{3,1}, \ldots, S_{3,k}, \ldots, S_{3,n})\right)$ in der Relation *Shadows* ebenfalls auch gelöscht wird.

5.8 Das gesamte Konzept für die Realisierung eines SRS

Werden alle Resultate zusammengefaßt, so können wir ein gesamtes Konzept für die Realisierung des Schlüssel-Rückgewinnung-Systems angeben. Die einzige Änderung ist: jeder Benutzer, der seine Schlüsselregistrieren will, ist ein Submitter. Dadurch kann die zweifach Verschlüsselung vermieden werden. Andererseits müssen die Benutzer nicht mehr ihre Geheimnisse in den Submitter eingeben, um dort mit dem Algorithmus *share* die Shares zu den Schlüsseln generieren zu lassen. Mit diesem Ansatz wird die Geheimnisse nicht mehr über das Netzwerk transportiert.

Allerdings gehört der Algorithmus *share* nicht mehr zum System. Dieser Algorithmus läuft auf jedem der μ Rechner des Benutzers. Um diesen Algorithmen kryptographisch zu schützen, muß der Sicherheitsadministrator μ-mal mit $SIG(\cdot)$ signieren. Jedesmal wenn der benutzer die Shares zu seinen Schlüssel generieren will, muß er mit

dem öffentlichen Verifikationsalgorithmus $VER(\cdot)$ die Authentizität des Algorithmus *share* überprüfen. Nach der überprüfung kann der Benutzer mit der Generierung der Shares beginnen.

5.9 Sicherheitspolitik nach der Rekonstruktion

Wenn bei der Einfügung der Schlüssel nicht hinreichend beschrieben wird, dann kann die Rekonstruktion weder dem Schlüsselbesitzer noch einem neuer Benutzer wenig helfen. Der neuer Benutzer kann beispielsweise einerseits nicht wissen, für welche Verschlüsselung der rekonstruierte Schlüssel ist. Andererseits weiß die Geschäftsleitung nicht, welchen Schlüssel für bestimmte verschlüsselte Daten rekonstruiert werden soll. Bei den Vorgeschlagenen Konzepten ist die Beschreibung des eingefügten Schlüssel offen und kann daher bei jeder Schlüsseleinfügung ausgenutzt werden. Aus organisatorischen Gründen sollen die Benutzer verpflichtet sein, die Beschreibung ihrer Schlüssel bei der Einfügung vornehmen. Beider Datenbank-Methode kann ein Benutzer seinen Schlüssel unter dem Attribut **Decription** in der Relation (Tabelle 5.2, Seite 124) *Secret* wie folgt beschreiben:

Decription 1 = Privater Schlüssel von Frank. Der Schlüssel für die Verchlüsselung der Bilddaten benutzt.

Diese Daten werden offenbar vom SRS nicht benötigt, um den Schlüssel zu rekonstruieren. Zusammen mit dem Schlüssel wird die *Decription 1* als Ergebnis der Rekonstruktion ausgegeben.
Frank kann natürlich falsche Besschreibung für seinen einzufügenden Schlüssel eingeben, so daß der Gebrauch des rekonstruierten Schlüssel für die anderen Verantwortlichen erschwert werden kann.

Wechselt Frank zu anderen Firma, so kann Industriespionage zustande kommen. Um zu verhindern, daß Frank geheime Daten allein entschlüsselt und sie zu der neuen Firma mitnimmt, kann die Firma den Franks privaten Schlüssel mit Hilfe des SRS rekonstruieren. Nach der Rekonstruktion werden die Daten entschlüsselt und der Schlüsselwert wird geändert. Der neuer Benutzer nimmt die Verantwortung auf diese Daten und kann das SRS anwenden, um seinen privaten Schlüssel registrieren zu lassen.

Diese Maßnahmen können das obige Problem nur relativ bzw. nicht wirksam lösen, denn Frank, abgesehen von seiner unverständlichen Beschreibung seines Schlüssels, kann die Daten schon vor seiner Kündigung heimlich, d.h. ohne Rekonstruktion, entschlüsselt und die Daten vorher mitgenommen hat.

Wir können das SRS geignet modifizieren und anwenden, um vor Industriespionage zu schützen. Man muß aber vorher die Bedeutung bzw. den Wert der geheimzuhaltenden Daten festsetzen. Danach kann man für eine Public-Key-Verschlüsselungsfunktion, beispielsweise RSA, festlegen, mit der diese Daten durch eine beliebige Person verschlüsselt wird. Bei der Entschlüsselung werden k von n Personen benötigt. In diesem fall wird die Entschlüsselungstransformation unter n Personen geteilt, so daß k

Personen die Daten entschlüsseln können. Weniger als k Personen ist die Entschlüng nicht möglich.
Gehöhrt zusätzlich Frank beispielsweise zur einer der autorisierten Teilmengen, so kann Frank allein die Daten nicht entschlüsselt.

Im Rahmen dieser Diplomarbeit können diese Probleme nicht gelöst werden. Weitere Informationen sind beispielsweise aus [11] zu finden. In diesem Artikel zeigen R. Gennaro, S. Jarecki, H. Krawczyk und Tal Rabin, wie die Signierung mit Hilfe der RSA-Funktion anhand des (k, n)-Threshold-Schemas geteilt wird.

Kapitel 6

Zusammenfassung und zukunftige Arbeiten

In dieser Arbeit wurden Konzepte für das Speicherungsproblem von Secret Sharing Schemata in der Praxis gemäß des (k, n)-Threshold-Schemas diskutiert, indem der Schlüsselspeicher eingeführt wird. Mit dem Schlüsselspeicher werden die Shares der eingefügten Schlüssel zentral gepeichert.

Wird dagegen die Speichermatrix $\mathbf{N}_E$ n Spaltenvektoren aufgespalten, so daß jeder Spaltenvektor an jeden Wächter verteilt wird, dann entsprechen die in der Arbeit diskutierten Konzepte dem Konzept des traditionellen Secret Sharing Schemas, bei dem die Shares zu den Schlüsseln lokal bei jedem Wächter gespeichert sind. Die ausgewählten Secret Sharing Schemata im Kapitel 4 können sehr gut angewandt werden, um die Betrugsprobleme elegant zu lösen. Zusätzlich müssen nur sichere Verschlüsselung, Signierung und Authentisierung eingesetzt werden, um die Geheimhaltung, Integrität der Shares und Verbindlichkeit im öffentlichen Netzwerk zu erreichen.

Die Nachteile dieser Verteilung sind der auftretende Kommunikationsaufwand zwischen Benutzern als Submitter und den Wächtern und die lokale Verwaltung der Shares auf jedem Wächter. Das Netzwerk muß somit hohe Bandbreite für jeden Knoten zur Verfügung stellen. Diese Anforderung ist aber nicht wirkschaftlich. Die Prüfverfahren im Kapitel 4 beanspruchen außerdem sehr viel Rechenkapazität, so daß alle Wächter Rechner mit hoher Rechenfähigkeit haben müssen. Das sind gerade die Probleme, welche wir in der Arbeit konzeptionell lösen wollen.

Durch Einfürung des Schlüsselspeichers werden die Shares für die Schlüsselregistrierung und -rückgewinnung nicht nur effizient gespeichert, sondern auch die Verwaltung der Shares viel einfacher durchgeführt werden. Die Speicherung der Shares kann mit bekannten Verfahren in der Informatik einfach implementiert werden. Bezüglich der verwendeten Speichertechnik wird die Sicherheit des Scghlüsselspeichers neben kryptographischen Tools erhöht, so daß die Rekonstruktion der eingefügten Schlüssel mit hoher Redundanz erfolgen kann. Mit dem gewählten (k, n)-Threshold-Schema von Shamir werden die Schlüssel der Mitarbeiter in einer Firma im Vergleich mit anderen Schutzmethoden vor Schlüsselkompromitierung besser geschützt.

Es wurde Absichtlich einen Submitter bei dem ersten konzept eingeführt, um die Resultate fur die Speicherung und Verwaltung der Shares im Schlüsselspeicher zu erzielen. Danach wird er bei dem zweiten Konzept ausgeblendet, um die Verwaltung der Benutzer und der Shares durch Datenbank-Konzepte zu beschreiben. Im dritten Konzept ist jeder Benutzer ein Submitter deklariert, damit die Eingabe der Schlüssel über Netzwerk zu vermeiden. Die Generierung der Shares bei jedem Submitter als Benutzer führt weiter dazu, daß die zweifache Verschlüsselung nicht mehr erforderlich ist. Allerdings müssen administrative Kontrollen verstarkt auf jedem Submitter durchgeführt werden, da jeder Benutzer einen Algorithmus *share* verfügt. Um die Sicherheit für die Erzeugung der Shares auf jedem Submitter muß der Algorithmus durch Sicherheitsadministrator signiert werden. Dadurch wird die Authentizität jedes Algorithmus *share* gesichert.

Die Computer-Hardware, insbesondere die Speichertechnologie, machen im Laufe der zeit riesigen Fortschritt. Fehlertolerante Computer werden gebaut. Doch kann man nicht ausschließen, daß das System fehlerfrei arbeitet. Obwohl der Schlüsselserver vor bösartigen Angriffen zusätzlich durch kryptographische Tools und vordefinierte Zugriffsoperationen geschützt wird, können die dort gespeicherten Shares durch einen Ausfall verloren gehen. Regelmäßige Sicherung des Schlüsselservers muß vorgenommen werden. Es mag sein, daß diese Art von Kopie die Sicherheit des SRS abschwächt. Doch die Grenzen zwischen Theorie und seine Implementierung können nicht beseitigt werden. In der Arbeit wurde gezeigt, daß das Threshold-Schema in der Tat uneingeschänkt sicher ist. Für praktische Anwendung ist die Sicherheit des Schemas aufgrund der Verschlüsselung und Signierung „nur“ berechnungssicher geworden.

Der Schlüsselspeicher wird dreifach geschützt. Gemäß des (k, n)-Threshold-Schemas können weniger als k Wächter keine Schüssel rekonstruieren. Durch Public-Key-Verschllüsselung können außer den Wächtern keine die Shares im Schlüsselspeicher aufdecken. Durch geeignete Speichertechnik dürfen nur die Wächter die Shares im Schlüsselspeicher lesen.

In dieser Diplomarbeit wurde das (k, n)-Threshold-Schema ausgewählt, um Konzepte für die Realisierung des Schlüsselspeichers, in dem die Gesamtheit der Shares der eingefügten kryptographischen Schlüssel gespeichert sind, zu demonstrieren. Wir haben auch das Schema von Benahlo und Leichter ausfülich beschrieben, mit dem die Rekonstruktion durch monotone Zugrifsstrukturen effizient realisiert wird.

Ob der Schlüsselspeicher gemäß monotone Zugriffstrukturen auch effizient implementiert werden kann, ist eine interessante Untersuchung.

In dem dritten Konzept wurden alle bisherigen Lösungsansätze zusammengefaßt. In diesem Konzept sollen alle Benutzer, die ihre Schlüssel registrieren wollen, als Submitter betrachtet. Der Algorithmus *share* mußen auf jeden Benutzer installiert werden. Damit trojanische Pferden nicht statt finden können, müssen alle dieser Algorithmen vom Sicherheitsafministrator signiert werden. Bei jeder Teilung des Schlüssels muß der Benutzer mit dem öffentlichen Verifikationsalgorithmus checken, damit die Authetizität des Algorithmus gewährleistet wird.

Wie diese Algorithmen signiert werden, ist eine sehr interessante Ergänzung der vorliegenden Arbeit.

Literaturverzeichnis

[1] J. Benalo and J. Leichter. Generalized secret sharing and monotone functions. *In „Advances in Cryptology – CRYPTO '88", S. Goldwasser, ed., Lecture Notes in Computer Science 403*, pages 27–35, 1989.

[2] G. R. Blakley. Safeguarding gryptographic keys. *In „Proceedings of The National Computer Conference, 1979", American Federation of Information Processing Societies Proceedings 48*, pages 313–317, 6-1979.

[3] C. Blundo, A. De Santis, and U. Vaccaro. On the information rate of secret sharing schemes. *(Preliminary version appeared in „Advances in Cryptology - CRYPTO '92") Theoretical Computer Science 154*, pages 283–306, 1996.

[4] R. M. Capocelli, A De Santis, L. Garnago, and U. Vaccaro. On the size of shares for secret sharing schemes. *(Preliminary version appeared in „Advances in Cryptology - CRYPTO '91")Journal of Cryptology 6*, pages 157–167, 1993.

[5] M. Carpentieri, A. De Santis, and U. Vaccaro. Size of shares and probability of cheating in threshold schemes. *In „Advances in Cryptology - EUROCRYPT '93", T. Helleseth, ed., Lecture Notes in Computer Science 765*, pages 118–125, 1994.

[6] D. Chaum. Computer system established, maintained, and trusted by mutually suspicious groups. *Memorandum N_0. UCB/ERL M79/10, University of California, Kerkeley, CA*, 2-1979.

[7] L. Czirimaz. The size of a share must be large. *In „Advances in Cryptology - EUROCRYPT '94", A. De Santis, ed., Lecture Notes in Computer Science 950*, pages 13–22, 1995.

[8] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, pages 77–94, 1988.

[9] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Advances in Cryptology, Crypto'86*, pages 186–194, 1987.

[10] National Institute for Standards and Technology (NIST). Digital signature standard (dss). *Technical Report 169. Ausgust 30 1991*, 1991.

[11] R. Gennaro, S. Jarecki, H. Krawczyk, and Tal Rabin. Robust and efficient sharing of rsa functions. *Crypto '96*, pages 157–172, 1996.

[12] L. Harn and H.-Y. Lin. An generalized secret sharing scheme with sheater detection. *Asiacrypt '91*, pages 149–158, 1991.

[13] F.P. Heider and D. Kraus und M. Welschenbach. *Mathematische Methoden der Kryptoanalyse*. Vieweg-Verlag, Wiesbaden, 1985.

[14] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. *Advances in Cryptology - CRYPTO '95"*, pages 339–352, 1995.

[15] M. Ito, A. Saito, and T. Nishiseki. Secret sharing scheme realizing general access structure. *In „Proceedings of the IEEE Global Telecommunications Conference, Clobecom '87",IEEE Press*, pages 99–102, 1987.

[16] E. D. Karmin, J. W. Greene, and M. E. Hellman. On secret sharing systems. *IEEE Transaction on Information Theory 29*, pages 35–41, 1983.

[17] L. Lamport. Password indentication with insecure communication. *Communication of the ACM*, pages 770–772, 1981.

[18] S. Langford. Threshold dss signatures without a trusted party. *In Crypto'95. Pringer Verlag. Lecture Notes in Computer Science N_o 963*, pages 397–409, 1995.

[19] T. Okamoto. Provably secure and practical identication and corresponding signature schemes. *Crypto '92*, pages 31–53, 1993.

[20] H. Ong, C. P. Schnorr, and A. Shamir. Efficient signature schemes based on polynomial equations. *Crypto '84*, pages 37–46, 1985.

[21] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. *Proc. of the 10-th ACM Symp. on the Princ. of Distr. Comp.*, pages 51–61, 1991.

[22] Alfred J. Menezes Paul, C. van Oorschot Scott, and A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[23] T.P. Pedersen. Distibuted provers with applications to undeniable signatures. *Advances in Cryptology - EUROCRYPT '91, Proceedings, Springer*, pages 221–242, 1991.

[24] T.P. Pedersen. Non-interactive and information-theoretic secure veriable secret sharing. *In „Advances in Cryptology - CRYPTO '91", Lecture Notes in Computer Science 576*, pages 129–140, 1991.

[25] A. Shamir. How to share a secret. *Communications of the ACM 22*, pages 612–613, 1979.

[26] C.E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, pages 656–715, 1948.

[27] Gustavus J. Simmons. How to (real) share a secret. *CRYPTO'88*, pages 391–448, 1988.

[28] E. Sperner. Ein stas über untermengen einer endlichen menge. *Math. Z.* , 27, pages 544–548, 1928.

[29] D. R. Stinson. New general lower bounds on the information rate of perfect sharing schemes. *In „Advances in Cryptology - CRYPTO '92"), E. F. Brickell, ed., Lecture Notes in Computer Science 740*, pages 168–182, 1993.

[30] Douglas R. Stinson. *CRYPTOGRAPHY - Theory and Practice.* CRC Press, 1995.

[31] D.R. Stinson. An explication of secret sharing schems. *Designs, Codes and Cryptography 2*, pages 357–390, 1992.

[32] M. Tompa and H. Woll. How to share the secret with cheater. *(Preliminary version appeared in „Advances in Cryptology - CRYPTO '86")Journal of Cryptology 1*, pages 133–138, 1988.

[33] H. Tzschach and G. Haßlinger. *Codes für den störungssicheren Datentranfer.* R. Oldenburg Verlag, 1993.

[34] CCITT Recomendation X.509. The diretory – authentication framework. *Consultation Committee, International Telefone and Telegraph, International Telecommunications Union, Geneva*, 1989.