9 Vertraulichkeit in Organisationen

Im militärischen, geheimdienstlichen und z.T. im (öffentlichen und privaten) Verwaltungsbereich wird traditionell gefordert, dass bestimmte **Geheimnisse** auf jeden Fall gewahrt bleiben, also bestimmte Dokumente geheim gehalten werden.

In den bisher vorgestellten Ansätzen (z.B. Oracle) verwalten die Eigentümer (Erzeuger) von Dokumenten die Rechte vollständig, insbesondere auch die Rechteweitergabe.
Jeder Eigentümer eines Dokuments kann in diesen Ansätzen Rechte nach eigenem Gutdünken weitergeben.

Aus der Sicht des militärischen Denkens wird dieser Ansatz daher auch **discretionary access control** (beliebige, ins freie Ermessen (des Eigentümers) gestellte Zugriffskontrolle) genannt.
Dieser Ansatz hat aus militärischer Sicht den Nachteil, dass man allen Eigentümern von Dokumenten trauen muss, dass sie (weder absichtlich noch unabsichtlich, z.B. durch trojanische Pferde) bei Rechteweitergabe-operationen nicht gegen die vorgegebene Sicherheitsstrategie verstoßen.
Ein militärisch-bürokratischer Ansatz legt Wert darauf, dass auch die indirekten (transitiven) Informationsflüsse gut überwachbar sind.

Wollte man z.B. Sicherheitsstufen mit einem discretionary access control Ansatz realisieren, so hieße dies konkret, dass der Eigentümer eines Dokuments, das als geheim eingestuft ist, sich überzeugen muss, das Leserecht nur Personen zu geben, deren Sicherheitsstufe mindestens geheim ist.
Stellt man sich vor, dass die Sicherheitsstufe eines Benutzers manchmal auch wieder kleiner gemacht werden muss (z.B. bei einem enttarnten Spion), so würde das gemäß der Sicherheitsstrategie viele Rechterückrufe erfordern, wozu viele Eigentümer von Dokumenten veranlasst werden müssten.

Der folgende Ansatz versucht daher, die Modellierung von festen, unüberwindbaren Grenzen für den Informationsfluss unmittelbar in den Zugriffskontrollmechanismus zu integrieren.
Er wird oft **mandatory access control** oder **nondiscretionary access control** (vorgeschriebene, unumgehbare Zugriffskontrolle) genannt.

Hier unterscheiden wir zunächst nicht-hierarchische Ansätze:
In der einfachsten Form ist dies eine **Zugriffskontrolle mit Abteilungsschranken** (*compartment access model*).
Außerdem stellen wir Verbreitungskontrollen und die Sicherheitspolitik der chinesischen Mauer vor.

Im nächsten Kapitel stellen wir dann hierarchische Ansätze zur mandatory access control vor.

Der in der Literatur am intensivsten untersuchte Fall, den wir hier auch ausführlicher behandeln wollen, ist das **Sicherheitsstufenmodell** (*multi-level security*).
Es wurde wesentlich aufgrund von militärischen und geheimdienstlichen Anforderungen entwickelt und wird inzwischen z.T. bei Beschaffungen im öffentlichen und militärischen Bereich vorgeschrieben.

••• **Gemeinsamkeit**: Kontrollmarkierung auf Sicherheitsetiketten (label), die untrennbar an das Granul geheftet werden. Umgekehrt werden auch Subjekte klassifiziert.

9.1 Das Zugriffsmodell mit Abteilungsschranken

Das Zugriffsmodell mit Abteilungsschranken ist ein Spezialfall der nondiscretionary access control.
Hier werden Abteilungsgrenzen als feste, unüberwindbare Grenzen für den Informationsfluss vorgegeben.
Im Idealfall implementieren diese Abteilungsschranken gerade das need-to-know-Prinzip.

Im einfachsten Fall gehören jedes Granul und jeder Benutzer genau einer Abteilung an.
Jeder Benutzer darf auf die Granule zugreifen, die zu seiner Abteilung gehören.
Dieser Ansatz lässt sich mit einem discretionary access control-Ansatz kombinieren.

Formal lässt sich dieser Ansatz beschreiben durch einen Menge von Abteilungen und zwei Funktionen, die Granulen bzw. Benutzern ihre Abteilungen zuordnen.

Folgende Erweiterungen sind denkbar:
1. Ein Benutzer darf mehreren Abteilungen angehören:
Solche Benutzer dürfen dann auf die Granule aller Abteilungen zugreifen, denen Sie angehören.
(Diese Idee geht später als Kategorien in das Bell-LaPadula-Modell ein.)

2. Ein Granul darf mehreren Abteilungen angehören:
Die naheliegende Interpretation wäre dann, dass auf ein solches Granul zugegriffen werden darf, wenn der Benutzer zu einer der Abteilungen gehört, zu der auch das Granul gehört.
Das führt aber dazu, dass transitive Informationsflüsse möglich sind:

Beispiel: Sei K={Produktion, Verwaltung, Verkauf},
klassifikation(g1)={Verwaltung, Verkauf},
klassifikation(s1)={Verwaltung},
klassifikation(s2)={Verkauf}.

Dann darf s1 auf g1 zugreifen (z.B. schreiben) und s2 auch auf g1 zugreifen (z.B. lesen).
Es wäre also ein transitiver Informationsfluss von s1 nach s2 über g1 möglich, obwohl die Klassifikationen von s1 und s2 einen leeren Durchschnitt haben.

Da man die Transitivität von Informationsflüssen in militärischen Modellen besonders gut überwachen möchte, ist diese Erweiterung nicht Bestandteil des Bell-LaPadula-Modells.

Transitive Informationsflüsse sind sowohl bei 1. und bei 2. möglich.
Das Problem bei 1. wird aber durch upward write / downward read gelöst.

9.2 Verbreitungskontrollen (*dissemination controls*)

Idee: Modellierung von alternativen nachrichtendienstlichen Geheimhaltungsstrategien

Jedes Granul bekommt eine **Kontrollmarkierung** auf einem **Etikett** (*label*), die den Personenkreis festlegt, für den der Dateninhalt freigegeben ist. Der Erzeuger des Granuls bringt die Markierung an.

Vorteil:
Kontrollmarkierungen können auf andere Benutzerattribute zurückgreifen als nur die Freigabe (*clearance*) des Benutzers. (mehr Semantik verfügbar)

Probleme:
Wie sollen Daten zusammengefügt werden, die unterschiedliche Kontrollmarkierungen tragen?
Kontrollmarkierungen können komplex werden!

Beispiel 1:

NOFORN (*not releasable to foreign nationals*): nicht freizugeben an Ausländer, Zugriff nur für US-Bürger.

REL <Staat oder Person> (*release for …*):
Das Dokument wird ausnahmsweise auch freigegeben für Person … oder Staatsangehörige des Staates ….

Also NOFORN/REL CAN: Zugriff für US-amerikanische und kanadische Staatsbürger.
NOFORN/REL James Bond: Zugriff für US-amerikanische Staatsbürger und Briten James Bond.

Die Kombination von Dokument1 (Markierung NOFORN/REL UK) mit Dokument2 (Markierung NOFORN/REL CAN) ergibt ein Dokument mit Markierung NOFORN.

Zur Realisierung dieser Kontrolle muss natürlich die Nationalität des Benutzers und ggf. sein Name bekannt sein.

Beispiel 2:
ORCON (*originator controlled release*) [Gr 89]: Freigabe bleibt der Kontrolle durch Urheber unterworfen.
Die Kontrollmarkierung enthält wieder eine Liste von Organisationen (diesmal nicht auf Nationalitäten eingeschränkt), deren Mitarbeiter auf das Dokument zugreifen dürfen.
Diese Markierung wird vom Erzeuger des Granules angelegt und kann nur vom Erzeuger geändert werden.
Die Erzeuger werden nicht personen-, sondern organisationsweise bestimmt.
Für die Änderung von Markierungen bestimmt jede Organisation einen (oder mehrere) Berechtigte.

Für ein Dokument des Verteidigungsministeriums erhält auch das Landwirtschaftsministerium Zugriffserlaubnis (vom Verteidigungsministerium).
Wenn ein Mitarbeiter des Staatsministeriums auch Zugriff erhalten möchte, so kann nur ein dafür besonders Berechtigter des Verteidigungsministeriums (und nicht etwa ein Mitarbeiter des Landwirtschaftministeriums) die Erlaubnis erteilen.

Hat das Staatsministerium Zugriffsrechte auf ein Dokument des Verteidigungsministeriums und ein Dokument des Landwirtschaftsministeriums, und macht es aus beiden ein neues, drittes Dokument, so haben Verteidigungsministerium und Landwirtschaftsministerium als Urheber mitzubestimmen, wer den Zugriff auf das neue Dokument erhalten soll und wer nicht, obwohl das Staatsministerium Eigentümer des neuen Dokuments ist. (Das wird auf Dauer ziemlich kompliziert.)

Anforderungen an einen Mechanismus zur Verbreitungskontrolle
Der Mechanismus muss **Benutzerattribute** (wie Nationalität, Arbeitgeber, zugehörige Organisation) mit den zugreifenden Subjekten (Prozessen), die für den Benutzer arbeiten, verbinden können.

Die Zugriffskontrolle muss auf Granul-Etiketten basieren, damit die Information nicht über nicht oder mangelhaft markierte Granule abfließen kann.

Die Zugriffskontrolle muss auch auf Subjekt-Etiketten basieren, damit nicht ein (indirektes) Kopieren über ein Subjekt in nicht oder mangelhaft markierte Granule möglich ist.

Es muss Mechanismen geben für die Ableitung korrekter Markierungen im Falle der Kombination von Daten, sowie zur Modellierung einer geteilten Verantwortung für die Zugriffserteilung.

Das Etikett eines Granules enthält Markierungen über
eine Urheber-Kennung,
Attribute des Benutzers, für die das Granul freigegeben ist (in Form einer Zugriffskontroll-Liste).

Das Etikett eines Subjekts enthält Markierungen über
Attribute des Benutzers (wie Nationalität, Arbeitgeber, zugehörige Organisation)

Problem der Überklassifizierung (Granul-Etiketten werden immer größer und komplexer)
Downgrader nötig

9.3 Sicherheitspolitik der chinesischen Mauer

Idee: Modellierung von kommerziellen Geheimhaltungsinteressen,
z.B. bei einer Marktanalyse für eine Bank (Wahrung der Vertraulichkeit bzgl. der Firmenklienten, d.h. der Analysator kann keine Firmen beraten, wenn er Pläne, Zustand etc. des Wettbewerbers als *insider* kennt.
Er kann aber weitere Firmen beraten, die nicht miteinander in Wettbewerb stehen.)

Datenzugriff wird also nicht beschränkt durch Attribute der gewünschten Daten (wie BLP), sondern durch die Daten, auf die das Subjekt bereits Zugriff hatte.
Zugriff wird nur erlaubt auf Information, die nicht im Konflikt steht zu Information, die das Subjekt schon besitzt, also **in Abhängigkeit des Vorwissens des Subjektes**.
Unterschieden wird hier nicht zwischen Sicherheitsstufen (wie im militärischen Modell), sondern zwischen Eigentümern von Information.

Politik:
Annahme: Jedes Datum betrifft nur eine Firma.
Für jede **Firma** fassen wir die Menge ihrer Daten (*objects*) zusammen.
Daten von Firmen (*company dataset*), die im Wettbewerb miteinander stehen, werden in die gleiche Klasse (*conflict of interest class*), **Branche** genannt, getan.
Jedes Subjekt darf dann auf höchstens eine Datenmenge pro Klasse zugreifen.

Beispiel: Beteiligte Branchen seien Bank und Ölfirma.
Firmen seien Bank-A, Ölfirma-A, Ölfirma-B.
Ein neues Subjekt S darf stets auf jedes Datum zugreifen.
(Denn da es bisher keine Information besitzt (zumindest aus Sicht des Computersystems), kann kein Konflikt existieren.)

Angenommen S greift zuerst auf Daten der Ölfirma-A zu (wir sagen, dann "besitzt es Information" über diese Firma) und möchte anschließend auf Daten der Bank-A zugreifen.
Da Ölfirma-A und Bank-A verschiedenen Branchen angehören, gibt es keinen Konflikt, der Zugriff ist erlaubt.
Will S dann auf Daten der Ölfirma-B zugreifen, so wird dieser Zugriff verhindert, da es einen (Interessens–)Konflikt zwischen Ölfirma-A und Ölfirma-B gibt.
Die Reihenfolge des Zugriffs ist wichtig: Hätte S zuerst auf Daten der Ölfirma-B zugegriffen, so wäre dies erlaubt worden, allerdings ein anschließender Zugriff auf Daten der Ölfirma-A verweigert worden.
Bei Zugriffen auf Firmendaten unterschiedlicher Branchen spielt die Reihenfolge allerdings keine Rolle.

Zu Beginn hat jedes Subjekt also die vollständige Freiheit, auf welche Daten es zugreifen möchte.
Jeder Zugriff schließt das Subjekt dann allerdings vom Zugriff auf Daten konkurrierender Firmen der gleichen Branche aus.

Die Daten werden also in drei Stufen gruppiert:
Branchen
Firmendaten
Datum

Implementierung:
Nötig ist eine Tabelle, die für jedes Subjekt festhält, auf Daten welcher Firma das Subjekt schon zugegriffen hat. (* Tabelle Subjekt-Firmen-Tabelle reicht, nicht Subjekt-Granul-Tabelle *)

Formalisierung:
Sei S Menge von Subjekten,
G Menge von Granulen,
F Menge von Firmen,
B Menge von Branchen.

Jedes Granul erhält ein **Sicherheitsetikett** (*label*), auf dem die Firma steht, dessen Daten es enthält:
$l : G \rightarrow F$ (*Firma als Etikett reicht; nicht Firma und Branche*)

Außerdem wird jede Firma eindeutig einer Branche zugeordnet:
$b: F \rightarrow B$

Vorwissen: $N: S \times F \rightarrow BOOL$; $N(s,f)=true :\Leftrightarrow$ Subjekt s hat auf Granul(e) der Firma f zugegriffen.

Anfangszustand: $s \in S, f \in F : N(s,f)=false$

Zugriff: Subjekt s darf auf Granul g zugreifen $:\Leftrightarrow \quad f' \in F: N(s,f')=true \quad l(g)=f' \quad b(l(g))\neq b(f')$.

(Das Granul g gehört zu den Daten einer Firma f', auf die Subjekt s schon zugegriffen hat, oder es gehört zu den Firmendaten einer Branche, zu der s noch keinen Zugriff hatte, oder es hat überhaupt noch kein Zugriff von s stattgefunden.)
Im Falle des Zugriffs wird $N(s,l(g)) := true$ gesetzt.

Eine Beratungsfirma, die alle Firmen beraten will, braucht also mindestens so viele Mitarbeiter, wie Firmen in der größten (nach Anzahl der Firmen berechnet) Branche sind.

Änderungen:
Neue Daten müssen einer Firma zugeordnet werden.
Neue Firmen müssen einer Branche zugeordnet werden.
Neue Branchen können erzeugt werden.

Werden bestehende Branchen aufgeteilt, so können einige Konflikte entfallen;
werden bestehende Branchen zusammengelegt, so können dadurch neue Konflikte entstehen!

Ungelöst:
Wann wird vergessen, dass ein Subjekt vor langer Zeit mal für eine Firma X tätig gewesen ist?

Erweiterungen:
Als öffentlich freigegebene Daten von Firmen (*sanitized information*), (z.B. Statistiken) können modelliert werden durch eine eigene Branche ("Statistik"), die nur eine einzige "Firma" enthält.
Dann kann jedes Subjekt auf diese Daten zugreifen.

9.4 Das Sicherheitsstufenmodell

9.4.1 Informelle Einführung

Die Forderung nach Geheimniswahrung wird umgesetzt, indem **Sicherheitsstufen**
(z.B. offen (=unklassifiziert) < vertraulich < geheim < streng geheim)
eingeführt werden und geeignet an Subjekte und Granule geheftet werden.

Jedem potentiellen Benutzer wird eine Sicherheitsstufe (oder **Freigabe**) (*clearance level*) zugeordnet.
Ein Subjekt ist i.a. ein Prozess, der für einen Benutzer ausgeführt wird.
Es hat eine Sicherheitsstufe, die von der Sicherheitsstufe des Benutzers abgeleitet wird (z.B. gleich der Sicherheitsstufe des Benutzers ist).
Jedem Dokument wird eine Sicherheitstufe (oder **Klassifikation**) (*classification level*) zugeordnet.
Ein Granul ist das maschinelle Abbild des Dokuments.

Schutz der Geheimnisse: Benutzer mit niedriger Sicherheitsstufe dürfen "nichts Geheimes lesen":

> Ein Subjekt darf dann ein Granul lesen, wenn seine Sicherheitsstufe größer oder gleich der Sicherheitsstufe des Granuls ist.

Schutz der Geheimnisse: Benutzer mit höherer Sicherheitsstufe dürfen Benutzern mit niedrigerer Sicherheitsstufe "nichts verraten":

> Ein Subjekt darf dann in ein Granul schreiben, wenn die Sicherheitsstufe des Granuls größer oder gleich seiner Sicherheitsstufe ist.

Sonst könnte ein Benutzer sein hoch klassifiziertes (z.B. gerade gelesenes) Wissen (absichtlich oder unabsichtlich) in ein Dokument schreiben, das dann wiederum von weniger hoch klassifizierten Benutzern gelesen werden könnte.
Unmittelbar angewendet führt diese Regel zu der (absurden) Situation, dass ein (hoch klassifizierter) Vorgesetzter seinen (niedriger klassifizierten) Untergebenen keine Befehle mehr erteilen darf, weil er mit Ihnen nur einseitig kommunizieren kann:
Er darf die Berichte seiner Untergebenen lesen, seinerseits Berichte an seine Vorgesetzten schreiben, aber keine Berichte oder Befehle an seine Untergebenen schreiben.

Daher fasst man die einem Benutzer zugeordnete Sicherheitsstufe als seine **maximale Sicherheitsstufe** auf, d.h. man ordnet jedem Benutzer ein Intervall von Sicherheitsstufen
[niedrigste Sicherheitsstufe des Systems .. maximale Sicherheitsstufe des Benutzers]
zu und ermöglicht dem Benutzer, seine **gegenwärtige Sicherheitsstufe** in diesem Intervall frei zu wählen.
••• bei totaler Ordnung : Intervall; bei partieller Ordnung: Teilverband •••

Jetzt kann ein Vorgesetzter die gleiche Sicherheitsstufe wählen wie seine Untergebenen und er kann dann wieder Befehle erteilen.
Das System liefert ihm in dieser Situation (allerdings auch nur) die gleiche Sicht, die seine Untergebenen von dem System haben.

Umgekehrt, möchte ein Benutzer mit höherer Sicherheitsstufe unbedingt in ein Dokument mit niedrigerer Sicherheitsstufe schreiben, so muss er die Sicherheitsstufe des Dokuments entsprechend erhöhen.

Ist dies erlaubt, so führt dies zu einer schleichenden Wanderung der Dokumente durch die Sicherheitsstufen hin zur höchsten Sicherheitsstufe.
Da ein Sicherheitssystem, in dem alles als streng geheim deklariert ist, sich von einem Sicherheitssystem, in dem alles als offen deklariert ist, nicht unterscheidet (sobald jemand überhaupt etwas mit einigen Daten machen muss, benötigt er die Sicherheitsstufe streng geheim!), ist dieser Zustand aus militärischer Sicht unerwünscht (auch von multi-level kann dann keine Rede mehr sein :-)).

Abhilfe schaffen können dann noch sog. **vertrauenswürdige Benutzer** (*trusted subjects*), denen außerhalb der bisher beschriebenen Sicherheitsstrategie das Recht gegeben wird, die Sicherheitsstufe von Granulen geeignet zu verkleinern.

Waren wir bei der Einführung des militärischen Ansatzes davon ausgegangen, dass wir dem Benutzer bei der Rechtevergabe lieber nicht ganz so viel vertrauen wollten (wie es beim discretionary access control Ansatz üblich ist), sind wir nun in einer Situation, wo wir (zumindest in eingeschränkter Form) genau wieder vertrauenswürdige Benutzer brauchen.

Außerdem hat jedes Subjekt und jedes Granul eine Menge von Kategorien.
Die Sicherheitsstufe (*security level*) ist zusammengesetzt aus (*classification level, set of categories*).

Praktisch sind natürlich folgende Fragen sehr relevant:
Wer ordnet nach welchen Kriterien den Benutzern Sicherheitsstufen zu?
Wer ordnet nach welchen Kriterien den Dokumenten Sicherheitsstufen zu?

Strategie gut durch folgende Überlegung festlegbar:
Was passiert, wenn man Dokumente verschiedener Sicherheitsstufen mischen will?

9.4.2 Formalisierung: Das Bell-LaPadula-Modell

Das Bell-LaPadula-Modell will Vertraulichkeit sicherstellen.
Es war Grundlage für die Implementierung eines MULTICS-Betriebssystemkerns.

Sei G eine Menge von **Granulen**,
S G eine Menge von **Subjekten**, (S G wird für die Erzeugungsoperationen benötigt)
O := {**r**ead, **e**xecute, **w**rite, **a**ppend} eine Menge von **Operationen**.
(append steht für eine reine Schreiboperation, write erlaubt Lesen und Schreiben auf einem Granul).

Sei S_T S eine Menge ausgezeichneter, besonders **vertrauenswürdiger Subjekte** (*trusted subjects*).

Sei weiter C eine Menge von **Klassifikationen**, die total geordnet ist,
und K eine Menge von **Kategorien**.
L := C × (K) ist dann die Menge der **Sicherheitsstufen** (*security level*).

Bemerkung: Auf L ist dann eine Ordnung induziert gemäß
(c,k) ≤ (c',k') :⇔ c≤c' und k k'
Sind C und K endlich, so ist L ein Verband.

Die Funktion f_s:S → L weist jedem Subjekt seine maximale Sicherheitsstufe (*clearance*) zu,
die Funktion f_g:G → L weist jedem Granul seine Sicherheitsstufe zu,
die Funktion f_c:S → L weist jedem Subjekt seine augenblickliche Sicherheitsstufe zu.

Jedes Subjekt darf nie mit einer höheren als seiner maximalen Sicherheitsstufe arbeiten, d.h. es gilt stets:
$s \quad S: f_c(s) \leq f_s(s)$.

Eine **BLP-Maschine** ist dann	BLP = (V, R, D) mit
Zustandmenge (*state set*)	V,
Eingabealphabet (*requests*)	R := {get, release, give, rescind, create, delete, change},
Ausgabealphabet (*decisions*)	D := {ja, nein, ? }.

Ein **Zustand** (*state*) $v \quad V$ ist ein 4-Tupel	v=(b,M,f,H) mit
gegenwärtige Zugriffsmenge (*current access set*) (welche Zugriffe finden gerade statt?)	$b \quad S \times O \times G$,
einer **Zugriffsrechtematrix** (*access matrix*)	$M : S \times G \rightarrow \quad (O)$,
einer **Zuordnung von Sicherheitsstufen** (*security level assignment*)	$f=(f_s,f_g,f_c)$,
und einer **Granulhierarchie**	$H : G \rightarrow \quad (G)$.

(Die Granulhierarchie entspricht im wesentlichen der Directory-Hierarchie des Betriebssystems.)

Ziel der Sicherheitspolitik ist es, dass Informationen nur von niedrigen zu höheren (oder gleich hohen) Sicherheitsstufen fließen können. (Davon ausgenommen sind nur die sog. vertrauenswürdigen Subjekte).

Dafür haben wir drei **Zustandsaxiome** (*security policy axioms*):

1. Wenn ein Subjekt ein Granul lesen darf, dann muss die maximale Sicherheitsstufe des Subjekts größer gleich der Sicherheitsstufe des Granuls sein. (*simple security property*)

v=(b,M,f,H) V: (s,r,g) b (s,w,g) b $f_g(g) \leq f_s(s)$

2. Für Subjekte, die nicht als besonders vertrauenswürdig ausgezeichnet sind, gilt:
Ein Granul kann nur von einem Subjekt gelesen werden, wenn die momentane Sicherheitsstufe des Subjektes größer gleich der Sicherheitsstufe des Granuls ist.
Ein Granul kann nur von einem Subjekt geschrieben werden, wenn die momentane Sicherheitsstufe des Subjektes kleiner gleich der Sicherheitsstufe des Granuls ist. (**-property, confinement property*)

v=(b,M,f,H) V:

a) (s,r,g) b s S_T $f_g(g) \leq f_c(s)$
b) (s,a,g) b s S_T $f_g(g) \geq f_c(s)$
c) (s,w,g) b s S_T $f_g(g) = f_c(s)$

3. Ein Subjekt darf nur dann auf ein Granul mit Operation o zugreifen, wenn die Zugriffsmatrix diesen Zugriff erlaubt. (*discretionary security property*)

v=(b,M,f,H) V: (s,o,g) b o M(s,g)

Zustandsänderungen:
Für Zustandsänderungen gibt es 11 Regeln $r_1, r_2, \ldots r_{11}$.

Jede Regel ist eine Funktion $r_i: R \times V \rightarrow D \times V$,

die für den gegebenen Zustand und einer Eingabe einen neuen Zustand und eine Ausgabe liefert.

Das ? wird ausgegeben, falls eine Regel nicht vorgesehen ist für eine Eingabe x: $r_i(x,v)=(?,v)$.

Jede Regel verweigert Eingaben, wenn sie dadurch das System in einen Zustand bringen würde, der die Sicherheitspolitik verletzt. In einem solchen Fall ist $r_i(x,v)=(no,v)$.

Wird die Eingabe akzeptiert, so liefert die Regel $r_i(x,v)=(yes,v')$ mit einem Folgezustand v'.

Für jede der folgenden Typen von Eingaben gibt es eine Regel:
1. get-read
2. get-append
3. get-execute
4. get-write

Es wird jeweils ein Element zu b hinzugefügt.
Die Konsistenz mit den drei Sicherheitsaxiomen wird überprüft.

5. release-read/execute/write/append
Es wird ein Element von b gelöscht.

6. give-read/execute/write/append
7. rescind-read/execute/write/append
Fügt eine Zugriffserlaubnis in M hinzu bzw. löscht eine Zugriffserlaubnis in M.
Diese Operationen haben zwei Subjektparameter: den Verursacher und den Betroffenen dieser Zustandsänderung.
Die Regeln überprüfen, dass der Verursacher einen Schreibzugriff auf das Vatergranul des Granuls hat, das von der Zugriffserlaubnis betroffen ist.

8. create granule
9. delete granule-group
Die benutzte Speicherbereiche entsprechen aktiven Granulen, nicht benutzte Speicherbereiche entsprechen inaktiven Granulen.

Durch diese Eingaben werden Granule dem aktiven Teil der Hierarchie zugeordnet oder daraus entfernt.
Eine create-Eingabe wählt ein (inaktives) Granul aus und fügt es als neues Kind eines spezifizierten Granuls (zu dem der Verursacher Schreibzugriff hat) in die Granulhierarchie H hinzu.
Die create-Regel sollte überprüfen, dass das neu hinzugefügte Granul inaktiv ist.
Insbesondere erhält das neu aktivierte Granul eine neue Sicherheitsstufe.
Ein inaktives (und damit auch ein neu erzeugtes) Granul sollte leer sein, d.h. keine Informationen enthalten.

Löschen eines Granuls entfernt dieses Granul aus der Hierarchie und macht auch alle Granule unterhalb dieses Granuls inaktiv.
Außerdem verlieren alle Subjekte den Zugriff auf diese Granule, d.h es werden die entsprechenden Elemente in b gelöscht.

Diese Regeln bewahren die sog. **Kompatibilitäts-Eigenschaft** (*compatibility*) der Hierarchie.
Eine Hierarchie ist kompatibel, wenn jedes untergeordnete Granul eine mindestens ebenso hohe Sicherheitsstufe hat wie sein Vater-Granul.
Diese Eigenschaft wird benötigt, um **versteckte Kanäle** (*covert channel*) zu verhindern:
Hätte sonst ein Granul eine niedrigere Sicherheitsstufe als sein Vater-Granul, dann könnte ein Subjekt mit der Sicherheitsstufe des Vater-Granuls das Granul löschen und diese Operation könnte ermittelt werden von einem Subjekt mit niedrigerer Sicherheitsstufe, welches Zugriff auf das gelöschte Granul hatte.

10. change-subject-current-security-level
11. change-granule-security-level
Die Regeln zur Änderung der Sicherheitsstufen ändern f_c bzw. f_g, lassen aber die gegenwärtigen Zugriffsmenge b konstant.
Es muss aber überprüft werden, dass der neue Zustand die Sicherheitsaxiome erfüllt.
Änderungen der Sicherheitsstufen sollten nur sehr bedachtsam erfolgen.
Insbesondere sollte das **Ruheprinzip** (*tranquility principle*) eingehalten werden:
Die Klassifikation eines aktiven Granuls darf nicht geändert werden während einer "normalen" Operation.

Erniedrigt man die Sicherheitsstufe eines Subjektes, so muss man annehmen, dass das Subjekt keinen Speicher hat oder alle lokalen Speicher des Subjekts gelöscht werden.

Selbst dann gibt es noch einen versteckten Kanal, da ein Subjekt in der gegenwärtigen Zugriffsmenge b die Information mit sich trägt, auf welche Granule es zugreifen darf.

Erhöht man die Sicherheitsstufe eines Granuls, bleibt das Problem, dass alle Granule mit der Zeit zur höchsten Sicherheitsstufe "hinwandern" und vertrauenswürdige Subjekte gebraucht werden, um die Sicherheitsstufe eines Granuls wieder auf ein "normal niedriges" Niveau zu bringen.

9.5 Integrität: Biba-Modell

Biba stellte fest, dass man den Stufenansatz leicht modifiziert auch für ein Modell verwenden kann, das Integrität sichert.

Auch hier werden Subjekten und Granulen sog. **Integritätsstufen** zugeordnet, z.B.

{lebenswichtig > sehr wichtig > wichtig > nicht so wichtig} oder

{verifiziertes Programm > getestetes Programm> ungetestetes Programm}.

Allerdings ist die Sicherheitsstrategie gerade umgekehrt wie beim Schutz von Vertraulichkeit:
Informationsfluss soll jetzt erlaubt sein, wenn das Ziel eine kleinere Integritätsstufe hat als die Quelle ("Information kann Integrität verlieren, aber nie gewinnen").
In diesem Modell können Subjekte Granule beobachten oder modifizieren und andere Subjekte aufrufen.

Die einfachste von Biba vorgeschlagene Zugriffskontrollstrategie, die **strikte Integritätsstrategie** (*strict integrity policy*) erlaubt einem Subjekt

- beobachtetenden Zugriff nur auf Granule mit einer Integritätsstufe größer oder gleich der des Subjektes,
- ändernden Zugriff nur auf Granule mit einer Integritätsstufe kleiner oder gleich der des Subjektes,
- aufrufenden Zugriff nur bzgl. Subjekte mit einer Integritätsstufe kleiner oder gleich der des aufrufenden Subjektes.

Das Ändern von Integritätsstufen ist bei dieser Strategie nicht erlaubt.

Variationen dieser Strategie sind:
Eine **low-water mark Strategie**: Hier kann ein Subjekt Granule beobachten, die eine kleinere Integritätsstufe haben, aber die eigene Integritätsstufe wird dadurch entsprechend reduziert.
(Dies modelliert eine Analogie zum Konzept "schlechten Umgang" im sozialen Bereich).

Eine **low-water mark Strategie für Granule**: Dies ist eine low-water mark Strategie, in der ein Subjekt auch Granule mit höherer Integritätsstufe ändern kann, aber die Integritätsstufe dieser Granule wird dann automatisch reduziert.

Eine **ring-Strategie**, in der Beobachtung keinen Einschränkungen unterworfen ist.

Die low water mark Strategien erfüllen die Zustandsaxiome für strikte Integrität, aber auf Kosten von Integritätsstufen-Änderungen. Die Ring-Strategie funktioniert nur, wenn man annehmen darf, dass ein Subjekt mit hoher Integrität ein Programm mit hoher Integrität ausführt, d.h. das Programm nicht dadurch irritiert wird, dass es Granule mit niedriger Integrität beobachtet. Die Ring-Strategie würde besser werden, wenn man ausführenden Zugriff von beobachtendem Zugriff unterscheiden könnte.

Berücksichtigt man nicht den aufrufenden Zugriff, so sind die verbleibenden Zugriffsbeschränkungen für strikte Integrität gerade das Duale der *-Property. Versteht man "beobachten" als "lesen" und "ändern" als "schreiben", bleibt als einziger Unterschied, dass die Richtung der partiellen Ordnungen umgekehrt sind. Daher kann man zur Durchsetzung von Vertraulichkeit und Integrität die gleichen Mechanismen verwenden. Hat man eine Menge der Sicherheitsstufen C und Menge der Integritätsstufen I, so könnte man mit den Stufen $L := C \times I$ arbeiten und der partiellen Ordnung
$(c,i) \leq (c',i') \quad :\Leftrightarrow \quad \text{if } c \leq c' \qquad i \geq i'.$

Das größte praktische Problem bei diesem Ansatz ist, herauszufinden, welche Integritätsstufen man gebrauchen möchte und was sie genau bedeuten. Insbesondere darf man die Menge der Sicherheitsstufen und die Menge der Integritätsstufen nicht miteinander identifizieren, da dann ein Subjekt nur noch auf Granule der gleichen Stufe zugreifen dürfte.

Auch hier fehlen "Prüfer" (als Analogie zu den vertrauenswürdigen Benutzern), die die Integrität eines Granuls (oder eines Subjektes) wieder heben können.

9.6 Datenbanksysteme mit Sicherheitsstufen

In Datenbanksystemen werden strukturierte Daten modelliert und abgespeichert.
Die folgende Frage stellt sich daher noch deutlicher als sonst:

- Welchen Dateneinheiten (einer wie großen Dateneinheit) wird eine Sicherheitsstufe zugeordnet?

Im **relationalen Datenmodell** bieten sich dafür an: Datenelement (Wert), Tupel, Attribut, Relation, Datenbank, Sicht.

9.6.1 Zuordnung von Sicherheitsstufen zu relationalen Dateneinheiten

Verschiedene Möglichkeiten sind erforscht worden:

1. Sicherheitsstufen werden **attributweise** zugeordnet (Hinke/Schäfer, 1975):
In diesem Ansatz werden folgende Axiome benötigt:
    1. Die Primärschlüsselattribute einer Relation haben die gleiche Sicherheitsstufe.
    2. Die Nichtprimärschlüsselattribute einer Relation haben eine Sicherheitstufe, die größer oder gleich der Sicherheitsstufe der Primärschlüsselattribute ist.

Dies liegt daran, dass das DBMS, um ein Datenelement zu lesen, zunächst das passende Tupel finden und dazu den Primärschlüssel lesen muss.
Eingabe und Änderung von Tupeln sind aber kompliziert, sobald eine Relation Attribute verschiedener Sicherheitsstufen hat, da ein Benutzer, der hochklassifizierte Attributwerte eingeben will, nicht auch niedrigklassifizierte Attributwerte eingeben kann und umgekehrt.
(Also zunächst mit niedriger Sicherheitsstufe Tupel erzeugen und Schlüsselattribute schreiben, danach mit höherer Sicherheitsstufe weitere Attributwerte einfügen …)

2. Sicherheitsstufen werden **relationsweise** zugeordnet (IP Sharp model, Grohn, 1976):
Diese Vorgehensweise löst das praktische Problem der Einfüge- und Änderungsoperationen unter 1. Darüberhinaus ist es durch vertikale Zerlegung der Relation (Partitionierung der Nichtschlüsselattribute nach Sicherheitsstufen) trotzdem möglich, die bisherigen attributweisen Sicherheitsstufen zu simulieren:

Beispiel: die Relation habe die Attribute:
K,L als Schlüsselattribute,
K, L, M seien als vertraulich , N,O als geheim, P,Q als streng geheim klassifiziert.

attributweise:

K	L	M	N	O	P	Q
vertraulich	vertraulich	vertraulich	geheim	geheim	streng geheim	streng geheim

Die Simulation im IP Sharp Modell bildet dann die Relationen:

geheime Relation:

K	L	N	O

streng geheime Relation:

K	L	P	Q

vertrauliche Relation:

K	L	M

Als Problem bleibt jedoch, dass der Benutzer die verschiedenen Relationen konsistent halten muss (indem er für (im ersten Ansatz) zusammengehörige Tupel die gleichen Schlüssel benutzt).

3. Sicherheitsstufen werden **datenelementweise** zugeordnet (Naval DBMS, Graubart/Woodward, 1982):
Tupel, Attribute, Relationen, Datenbank werden als Container betrachtet und können eine eigene Sicherheitsstufen-Vorbesetzung (default security level, DSL) haben.
Die aktuelle Sicherheitsstufe eines Datenelements ergibt sich aus der DSL des Datenelements und der DSL aller Container, in denen das Datenelement enthalten ist.

•••Wie? (nun, z.B. als das Maximum aller umfassenden Container)

4. Sicherheitsstufen werden **Sichten** zugeordnet (Denning, 1987):
Dies erlaubt eine hohe Flexibilität:
Die Sicherheitsstufen können je nach Formulierung der Sicht den verschiedenen Ebenen (datenelementweise, attributweise etc.) zugeordnet werden.
Es können auch Aggregationen geschützt werden.

9.6.2 Semantische (Integritäts-)Bedingungen und Sicherheitsstufen

Hat man nur Daten gleicher Sicherheitsstufe, so lassen sich die semantischen Bedingungen von einem Subjekt, das mit der gleichen Sicherheitsstufe arbeitet, ähnlich durchsetzen wie ohne Sicherheitsstufen:
Das Subjekt darf alle Daten lesen und schreiben, um die semantische Bedingung überprüfen zu können.

Hat man Daten unterschiedlicher Sicherheitsstufen, so können Komplikationen auftreten:
Ist die Überprüfung der Bedingung erlaubt (i.a. eine Leseoperation) ?
Ist eine evtl. Folgeänderung (i.a. eine Schreiboperation) erlaubt ?
Darf eine Fehlermeldung überhaupt dem Subjekt gegeben werden (aus Sicht des Benutzers eine Leseoperation) ?

Dürfen -und wenn nur zur Optimierung- bestimmte Zugriffsstrukturen verwendet werden (auf semantischen Bedingungen wie Schlüsselbedingungen bauen häufig Zugriffsstrukturen auf)?

9.6.3 Verträglichkeit semantischer Bedingungen mit Sicherheitstufen der Granule

An Verträglichkeitsbedingungen wurden z.B. vorgeschlagen:

a) Die Sicherheitstufen der Schlüsselwerte eines Tupels müssen gleich sein
(damit der volle oder kein Schlüssel zugänglich ist; aus Benutzersicht keine Nullwerte im Schlüssel!;
entity integrity: Kein Tupel darf Nullwerte im Primärschlüsselattribut enthalten).
Sonst ständen aus Sicht eines Benutzers mit niedriger Sicherheitsstufe Nullwerte in einigen Primärschlüsselattributwerten!
Z.B. hätten wir eine Relation mit Schema (A,B,C), in der A und B den Primärschlüssel bilden, so würde ein Benutzer mit Sicherheitsstufe geheim das Tupel

(10, geheim, X, streng geheim, 17, streng geheim)

als (10, null, null) lesen.

b) Die Sicherheitsstufen der Werte eines Tupels müssen größer gleich der Sicherheitsstufe der Schlüsselwerte des Tupels sein
(damit ein Zugriff auf das Tupel mit Hilfe des Schlüssels möglich ist;
entity integrity: Kein Tupel darf Nullwerte im Primärschlüsselattribut enthalten).
Sonst ständen aus Sicht eines Benutzers mit niedriger Sicherheitsstufe nur Nullwerte im Schlüssel!
Z.B. würde (Schema wie oben) ein Benutzer mit Sicherheitsstufe geheim das Tupel

(20, streng geheim, Y, streng geheim, 34, geheim)

als (null, null, 34) lesen.

c) Ist R[X] S[Y] eine Inklusionsabhängigkeit, so müssen die Sicherheitsstufen der X-Werte eines Tupels aus der Relation R größer gleich der Sicherheitsstufen der entsprechenden Y-Werte der entsprechenden Tupel in der Relation S sein (damit keine aus Sicht des Benutzers "hängenden Referenzen" entstehen; referential integrity).
Bilden speziell die Attribute aus X in R einen Fremdschlüssel für die Relation S, so müssen außerdem die Sicherheitsstufen der X-Werte eines Tupels aus R alle gleich sein. (wie a))

d) Die Sicherheitsstufe des Relationschemas (insbesondere der Relationname und die Attributtypen) muss kleiner gleich den Sicherheitsstufen der Tupel dieser Relation sein (damit das Datenwörterbuch für Zugriffe genutzt werden kann).

e) Die Sicherheitsstufe eines Sichtnamens muss größer gleich der Sicherheitsstufe der benutzten Basisrelationen sein (damit die Sichtrelation tätsächlich berechnet werden darf).

Zur Vereinfachung der Implementierung fordert man manchmal (z.B. SeaView-Projekt):

f) Die Sicherheitsstufen aller Tupel und Werte innerhalb einer Basisrelation sollen gleich sein (dadurch werden die Schwierigkeiten von der Ebene der Basisrelation auf die Ebene der Sicht verlagert).

9.6.4 Polyinstantiierung (*polyinstantiation*)

Es kann vorkommen, dass ein Benutzer ein Granul erzeugen (ändern) will mit dem gleichen Namen (gleichen Schlüsselattributen) wie ein schon vorhandenes, für ihn unsichtbares Granul.
Das unsichtbare Granul darf nicht überschrieben werden (es hat eine höhere Sicherheitsstufe!).

Will man die Existenz dieser Dateneinheit mit höherer Sicherheitsstufe nicht preisgeben, d.h. den Benutzer nicht über diesen Namenskonflikt aufklären, so muss man mehrere Versionen dieser Dateneinheit angelegen (maximal für jede Sicherheitsstufe eine).
Identifikatoren sind dann nur noch eindeutig bzgl. einer Sicherheitsstufe bzw. -umgekehrt gesehen- die Sicherheitsstufe wird ein (für Benutzer unsichtbarer) Bestandteil der Identifikatoren.

Insbesondere wenn mehrere Kategorien verwendet werden, kann die Anzahl der Sicherheitsstufen recht groß werden, so dass die Datenbank in Gefahr ist, fast ausschließlich Daten zu enthalten, die aus Sicht der obersten Sicherheitsstufe falsch sind ("Datenmüll").

Man unterscheidet zwischen polyinstantiierten Tupeln und polyinstantiierten Elementen.

- **Polyinstantiierte Tupel** sind Tupel, die alle die gleichen Werte für die Primärschlüsselattribute haben, allerdings unterschiedliche Sicherheitsstufen für diese Schlüsselattributte.
- **Polyinstantiierte Elemente / Werte** gibt es dann, wenn ein Nichtschlüsselattribut eines Tupels verschiedene Werte bzgl. unterschiedlicher Sicherheitsstufen annehmen soll.

Polyinstantiierte Tupel treten auf, wenn ein Tupel eingefügt wird, das den gleichen Primärschlüsselwert, aber eine niedrigere Sicherheitsstufe wie ein schon existierendes Tupel hat.
(Bei einer höheren oder gleich hohen Sicherheitsstufe würde man i.a. das alte Tupel ersetzen wollen.)

Das Ergebnis ist ein zweites Tupel in der Relation, dessen Primärschlüssel sich nur durch den Wert der Sicherheitsstufe unterscheidet.
Für den Verursacher der Änderungsoperation ist diese Polyinstanz zwar unsichtbar, aber Benutzer mit höherer Sicherheitsstufe können beide Tupel sehen.

Ein polyinstantiiertes Element tritt auf, wenn ein Wert geändert wird, der als Nullwert in der Relation erscheint, in Wirklichkeit jedoch ein (versteckter) Wert einer höheren Sicherheitsstufe ist.
(Eine weitere Möglichkeit wäre, wenn ein Benutzer mit hoher Klassifikation einen Wert mit niedriger Sicherheitsstufe ändert, jedoch verbieten die meisten System dies und liefern nur eine Fehlermeldung.)

Im Sinne des Entity-Relationship-Modells repräsentieren polyinstantiierte Elemente dasselbe Entity (der realen Welt), während polyinstantiierte Tupel verschiedene Entities (der realen Welt) repräsentieren.

In nicht-polyinstantiierten Relationen gilt, dass es zu einem Primärschlüsselwert (K) genau einen Wert für die Nichtprimärschlüsselattribute (A_i) gibt: K -> A_i.

In polyinstantiierten Relationen gilt, dass jedes Nichtschlüsselattribut A_i funktional abhängt außer von den Schlüsselattributen (K) auch noch von der Sicherheitsstufe des Schlüssels (C_K) sowie der Sicherheitsstufe des Nichtschlüsselattributs (C_i): K C_K C_i -> A_i.

Beispiel:
Eine Relation mit elementweiser Sicherheitsstufenzuordnung
(Jedes Attribut A erhält ein zusätzliches Klassifikationsattribut C). Sei A1 der Schlüssel.

A1	C1	A2	C2	A3	C3
mad	geheim	17	geheim	x	geheim
foo	geheim	34	geheim	w	streng geheim
ark	streng geheim	5	streng geheim	y	streng geheim

Die Sicht eines Benutzers mit Sicherheitsstufe "geheim" auf diese Relation:

A1	C1	A2	C2	A3	C3
mad	geheim	17	geheim	x	geheim
foo	geheim	34	geheim	null	geheim

Die Sicherheitsstufe des Nullwertes sollte die Sicherheitsstufe des Schlüssels haben.

Das Einfügen eines Tupels mit den gleichen Primärschlüsselwerten, aber niedrigerer Sicherheitsstufe, erzeugt ein polyinstantiiertes Tupel (4.Tupel):

A1	C1	A2	C2	A3	C3
mad	geheim	17	geheim	x	geheim
foo	geheim	34	geheim	w	streng geheim
ark	streng geheim	5	streng geheim	y	streng geheim
ark	geheim	22	geheim	z	geheim

Der Benutzer mit Sicherheitsstufe geheim ergänzt (foo,34,null) um einen A3-Wert und erzeugt so ein polyinstantiiertes Element/Wert (3.Tupel):

A1	C1		A2	C2		A3	C3	
mad	geheim		17	geheim		x	geheim	
foo	geheim		34	geheim		w	streng	geheim
foo	geheim		34	geheim		u	geheim	
ark	streng	geheim	5	streng	geheim	y	streng	geheim

••• manchmal Tupelklasse TC interessant

••• absichtliche Polyinstantiierung: Cover Stories