An act of sabotage can have two principal effects: improper functioning of the system and incorrect semantic content. An integrity policy attempts to prevent acts of sabotage within the information system or to localize the effects to an acceptable degree.

Work on a model or integrity policy implementation is proceeding at MITRE [23]. A major problem is to specify an acceptable and appropriate policy to govern the modification of data segments. We consider here a simple model of integrity, leaving policy largely unspecified, in order to further the exposition of the problem.

Suppose that a set \( S \) of "integrity levels" is given: consider as an example the set:

nonsensitive < sensitive < critical < very critical

The semantics of these terms is suggestive; the integrity policy is, nevertheless, not specified by them since they are not formally defined. Suppose further that integrity level functions, analogous to security level functions, are defined:

\[
I_S: \{\text{subjects}\} \rightarrow \{\text{integrity levels}\} \quad \text{and} \quad I_O: \{\text{objects}\} \rightarrow \{\text{integrity levels}\}.
\]

\( I_S(\text{subject}) \) denotes the maximum integrity level of an object that \text{subject} is allowed to modify; \( I_O(\text{object}) \) denotes the minimum level of any subject that is allowed to modify \text{object}.

Redefine a state \( v \) of the system by the inclusion of \( I = (I_S, I_O) \):
$v = (b, M, f, I, H)$.

We can define a **simple-integrity-property** (si-property), analogous to the ss-property, as follows:

A state satisfies the si-property provided for every current alter-access (subject, object, alter-attribute), the integrity level of subject ($I_S(\text{subject})$) is greater than or equal to the integrity level of object ($I_O(\text{object})$).

More formally, $v = (b, M, f, I, H)$ satisfies the si-property provided:

$$[(S_i, O_j, x) \text{ in } b \text{ and } x \in \{w, a\}]$$

implies $I_S(S_i) \geq I_O(O_j)$.

There is an alternative formulation of the si-property, as there is for the ss-property:

The state $v = (b, M, f, I, H)$ satisfies the si-property provided every $(S_i, O_j, x)$ in $b$ satisfies the **simple-integrity condition relative to I** (SIC rel I); $(S_i, O_j, x)$ in $b$ satisfies SIC rel I provided ($x = w$ or $x = a$) implies that $I_S(S_i) \geq I_O(O_j)$.

Given the above extension of the model, needed modifications to the rules of operation are obvious; moreover, intuition indicates that assuring the si-property systematically is inductive and can be accomplished by demonstrating si-property preservation over one state change (as is the case for secure state preservation). No analogue to the **-property exists**, since the problem of information transfer within the realm of disclosure has no analogue in the
realm of sabotage. Finally, an inverse compatibility property for the hierarchy seems attractive; this would dictate that the integrity level of objects be monotone non-increasing on paths away from the root. This latter property relates to "localizing" damaging effects of sabotage action. Actual sabotage of sensitive-directory in Figure 18 indirectly sabotages inferior segments, which are necessarily nonsensitive or sensitive under inverse compatibility; the effect of sabotaging sensitive-directory by a sensitive process running amok would not extend to its parent, critical-directory, nor to unrelated segments such as critical-segment, sensitive-segment, and nonsensitive-segment.

Figure 18. The Subtree Affected by Sabotage of Sensitive-Directory
APPENDIX

Introduction

The formal mathematical model is presented in this Appendix. No interpretation or explanation is offered, except as subsequently noted. The intended interpretations and correspondences to Multics architectural elements are given in the body of this report. In the section of this Appendix on rules, a narrative statement of each rule is given in order to reduce the reader's inconvenience in dealing with highly abstract symbology and in order to provide a natural language statement of intention by which errors or policy misdirections in the formal statements may be more easily discovered.

Elements

The elements of the mathematical model are presented in Table I. Some items are not self-explanatory and they are explained here.

**partial ordering relation \( \succ \):**

A relation \( R \) is a partial ordering relation if \( R \) is reflexive, antisymmetric, and transitive.

Suppose that \( U \) is a set and \( R \) is a binary relation defined on \( U \), with elements of \( U \) denoted by small letters \( a, b, c, \ldots \) etc.

**reflexive:** \( R \) is reflexive if \( xRx \) for each \( x \) in \( U \).

**antisymmetric:** \( R \) is antisymmetric if \([xRy \text{ and } yRx] \) implies
\[ x = y \ (x \text{ is identically } y) \text{ for each } x \text{ and } y \text{ in } U. \]
(In other words, we have \( xRy \) and \( yRx \) (symmetry) only in case \( x = y \).)

**transitive:** \( R \) is transitive if \([xRy \text{ and } yRz]\) implies \( xRz \) for each \( x \) and \( y \) and \( z \) in \( U \).

\[ L = \{L_1, L_2, \ldots, L_p\} \text{ where } L_i = (C_j, K) \text{ and } C_j \text{ is in } \]
\( C \) and \( K \) is a subset of \( K \). Define the relation \( \sim \) on \( L \) as follows:

\[ (L_i, L_j) \in \sim \equiv L_i \sim L_j \equiv (C_i, K) \sim (C_j, K') \text{ iff} \]

(i) \( C_i \geq C_j \), and

(ii) \( K \supseteq K' \).

Since both "\( \geq \)" and "\( \supseteq \)" are partial orderings, a straightforward argument shows that "\( \sim \)" is also a partial ordering.

Suppose \( C = \{S, C, U\}, \ S \succ C \succ U, \) and \( K = \{K_1, K_2, K_3\} \) and \( L_1 = (S, \{K_1, K_2\}), \ L_2 = (S, \{K_1\}), \ L_3 = (C, \{K_1, K_2\}), \ L_4 = (C, \{K_1\}), \ L_5 = (S, \{K_2, K_3\}), \ L_6 = (C, \{K_2\}), \) and \( L_7 = (U, \{K_1\}) \). The partial ordering of these elements of \( L \) is illustrated as a digraph in Figure A1. L

\[ \text{Figure A1: Illustration of } \sim. \]
<table>
<thead>
<tr>
<th>SET</th>
<th>ELEMENTS</th>
<th>SEMANTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>${S_1, S_2, \ldots, S_n}$</td>
<td>subjects: processes, programs in execution</td>
</tr>
<tr>
<td>$O$</td>
<td>${O_1, O_2, \ldots, O_m}$</td>
<td>objects: data, files, programs, I/O devices</td>
</tr>
<tr>
<td>$C$</td>
<td>${C_1, C_2, \ldots, C_q}$</td>
<td>classifications: clearance level of a subject, classification of an object</td>
</tr>
<tr>
<td>$K$</td>
<td>${K_1, K_2, \ldots, K_r}$</td>
<td>categories: special access privileges</td>
</tr>
</tbody>
</table>

$\{L_1, L_2, \ldots, L_p\}$ with partial ordering relation $\preceq$, where $C_j \in C_i$ is in $C_i$ and $K$ is a subset of $K$
<table>
<thead>
<tr>
<th>SET</th>
<th>ELEMENTS</th>
<th>SEMANTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(r, e, w, a)</td>
<td>access attributes: r: read-only; e: execute (no read, no write); w: write (read and write); a: append (write-only)</td>
</tr>
<tr>
<td>RA</td>
<td>{g, r}</td>
<td>request elements: g: get, give r: release, rescind</td>
</tr>
<tr>
<td>S'</td>
<td>a subset of S</td>
<td>subjects subject to *-property:</td>
</tr>
<tr>
<td>S_T</td>
<td>S - S'</td>
<td>trusted subjects: subjects not subject to *-property but 'trusted' not to violate security with respect to it.</td>
</tr>
<tr>
<td>SET</td>
<td>ELEMENTS</td>
<td>SEMANTICS</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>R</td>
<td>$\bigcup_{1 \leq i \leq 5} R^{(i)}$, where</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(1)} = RA \times S \times O \times A$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(2)} = S \times RA \times S \times O \times A$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(3)} = RA \times S \times O \times L$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(4)} = S \times O$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(5)} = S \times L$</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>requests:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(1)}$: requests for get- and release-access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(2)}$: requests for give- and rescind-access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(3)}$: requests for generation and reclassification of objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(4)}$: requests for destruction of objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R^{(5)}$: requests for changing security level</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>{yes, no, error, ?}; an arbitrary element of D is written $D_m$</td>
<td><strong>decisions:</strong></td>
</tr>
</tbody>
</table>
Table I (Cont.)

<table>
<thead>
<tr>
<th>SET</th>
<th>ELEMENTS</th>
<th>SEMANTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>{1, 2, \ldots, t, \ldots}</td>
<td>indices: elements of a time set; identification of discrete moments; an element t is an index to request, decision, and state sequences</td>
</tr>
<tr>
<td>F</td>
<td>an element $f = (f_S, f_0, f_c)$ is in $F \subseteq L^S \times L^0 \times L^S$ if and only if for each $S_i$ in $S$ $f_S(S_i) \gg f_c(S_i)$</td>
<td>security level vectors: $f_S$: subject security level function $f_0$: object security level function $f_c$: current security level function</td>
</tr>
<tr>
<td>X</td>
<td>$R^T$; an arbitrary element of $X$ is written $x$</td>
<td>request sequences:</td>
</tr>
<tr>
<td>Y</td>
<td>$D^T$; an arbitrary element of $Y$ is written $y$</td>
<td>decision sequences:</td>
</tr>
<tr>
<td>SET</td>
<td>ELEMENTS</td>
<td>SEMANTICS</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>( M )</td>
<td>( {M_1, M_2, \ldots, M_{nm24}}; an element of ( M ), say ( M_k ), is an ( nxm ) matrix with entries from PA; the ( (i,j) ) - entry of ( M_k ) shows ( S_i )'s attributes relative to ( O_j ); the entry is denoted by ( M_{ij} )</td>
<td>access matrices: current access-permission structure; embodiment of discretionary security</td>
</tr>
<tr>
<td>( H )</td>
<td>an element ( H ) is in ( H \subseteq (PO)^0 ) if and only if: (1) ( O_i \neq O_j ) implies ( H(O_i) \cap H(O_j) = \emptyset ) (2) there does not exist a set ( {O_1, O_2, \ldots, O_w} ) of objects such that ( O_{r+1} ) is in ( H(O_r) ) for each ( r, 1 \leq r \leq w ), and ( O_{w+1} = O_1 )</td>
<td>hierarchies: a hierarchy is a forest possibly with stumps, i.e., a hierarchy can be represented by a collection of rooted, directed trees and isolated points.</td>
</tr>
<tr>
<td>( \text{tree}_H )</td>
<td>( \bigcup H(O) \cup H^{-1}(PO-(\emptyset)) )</td>
<td>the &quot;forest part of the hierarchy: if the hierarchy has a single tree, then ( \text{tree}_H ) can be represented by a single rooted, directed tree.</td>
</tr>
<tr>
<td>SET</td>
<td>ELEMENTS</td>
<td>SEMANTICS</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>grass</td>
<td>{system-wide variables} U</td>
<td>miscellanies:</td>
</tr>
<tr>
<td></td>
<td>{non-forest I/O devices} U</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{any other non-forest objects}</td>
<td></td>
</tr>
<tr>
<td>A(H)</td>
<td>$\text{tree}_H \cup \text{grass}$</td>
<td>the active objects:</td>
</tr>
<tr>
<td>B</td>
<td>$P(S \times O \times A)$; an arbitrary element of B is</td>
<td>current access set:</td>
</tr>
<tr>
<td></td>
<td>written $b$</td>
<td>record of current access of subjects to objects in various modes</td>
</tr>
<tr>
<td>V</td>
<td>$B \times I \times F \times H$; an arbitrary element of V</td>
<td>states:</td>
</tr>
<tr>
<td></td>
<td>is written $v$</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>$V_T$; an arbitrary element of $Z$ is written $z$; $z_t$ in</td>
<td>state sequences:</td>
</tr>
<tr>
<td></td>
<td>$z$ is the $t$-th state in the state sequence $z$</td>
<td></td>
</tr>
</tbody>
</table>
Suppose \([U, R]\) is a partially ordered system. An element \(m\) in \(U\) is called a **minimal element** in \(U\) if \(mRx\) implies \(xRm\) for each \(x\) in \(U\); if \(m\) is unique it is called a **minimum**. For \([L, \infty]\), as in the previous example, there are three minimal elements, \((U, K_1), (U, K_2),\) and \((U, K_3)\) and there is no minimum. If \(K' = K \cup \{\phi\}\), then \((U, \phi)\) is a minimum in \([C \times K', \infty]\).

**The notation \(A^B\):**

Suppose \(A\) and \(B\) are sets. The notation \(A^B\) denotes the set of all functions from \(B\) to \(A\). Suppose \(A = \{a, b\}\) and \(B = \{1, 2\}\) then \(A^B\) consists of

\[f_1 = \{(1, a), (2, b)\},\]
\[f_2 = \{(1, b), (2, a)\},\]
\[f_3 = \{(1, a), (2, a)\}, \text{ and}\]
\[f_4 = \{(1, b), (2, b)\}.

**Cartesian Product:**

Suppose \(A\) and \(B\) are sets. The **cartesian product** of \(A\) and \(B\), denoted \(A \times B\), is defined by

\[A \times B = \{(a, b): a \in A \text{ and } b \in B\},\]

i.e., \(A \times B\) is the set of all ordered pairs of the form \((a, b)\) where \(a\) is in \(A\) and \(b\) is in \(B\). Suppose \(A = \{a, b\}\) and \(B = \{1, 2\}\). Then \(A \times B = \{(a, 1), (a, 2), (b, 1), (b, 2)\}\). Notice that \(B \times A = \{(1, a), (2, a), (1, b), (2, b)\}\) ≠ \(A \times B\). Notice also that \(f_1 \subseteq B \times A\), \(f_1\) defined above.
the notation \( PX \):

Suppose \( X \) is a set, say \( X = \{a, b, c\} \). \( PX \) means the set of all subsets of \( X \). In this case, \( PX = \{\phi, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\} \) where \( \phi \) denotes the empty set.

hierarchies:

Suppose \( H \subseteq (PO)^0 \) where \( O = \{O_1, O_2, O_3, O_4, O_5\} \). Restrict membership in \( H \) by the conditions (1) and (2) (see Table 1, entry for \( H \)). Define \( H \in H \) as follows:

\[
H = \{(O_1, \{O_2, O_3\}), (O_2, \phi), (O_3, \{O_4, O_5\}), (O_4, \phi), (O_5, \phi)\}.
\]

\( H \) can be described also by a diagraph:

```
\begin{center}
\begin{tikzpicture}[scale=.7]
    \node (O1) at (0,4) {$O_1$};
    \node (O2) at (-2,2) {$O_2$};
    \node (O3) at (2,2) {$O_3$};
    \node (O4) at (-2,-2) {$O_4$};
    \node (O5) at (2,-2) {$O_5$};
    \path[->, thick]
    (O1) edge node {} (O2)
    (O1) edge node {} (O3)
    (O2) edge node {} (O4)
    (O3) edge node {} (O5);
\end{tikzpicture}
\end{center}
```

Condition (1) rules out a structure such as:

```
\begin{center}
\begin{tikzpicture}[scale=.7]
    \node (O1) at (0,4) {$O_1$};
    \node (O2) at (-2,2) {$O_2$};
    \node (O3) at (2,2) {$O_3$};
    \path[->, thick]
    (O1) edge node {} (O2)
    (O1) edge node {} (O3)
    (O2) edge node {} (O3);
    \node (X) at (0,0) {$\times$};
\end{tikzpicture}
\end{center}
```
and condition (2) rules out a structure such as

If an element of \( H \) imposes a forest structure on the objects with exactly one tree, as in the example, we identify the root of the tree by the notation \( O_R \). If \( H \) is a tree structure then \( O_R \) is that object in \( O \) for which

\[
H(O_R) \neq \emptyset \text{ and } O_R \notin H(O) \text{ for any } O \in O.
\]

If \( O_j \) is an object in \( O \) then \( O_s(j) \) denotes that object with respect to \( H \) such that \( O_j \in H(O_s(j)) \); in other words \( O_s(j) \) is "superior" to \( O_j \) by \( H \).

**System**

Suppose that \( W \subseteq R \times D \times V \times V \). The **system**\( \Sigma \) \((R, D, W, z_0) \subseteq X \times Y \times Z \) is defined by

\[
(x, y, z) \in \Sigma(R, D, W, z_0) \text{ iff } (x_t, y_t, z_t, z_{t-1}) \in W \text{ for each } t \text{ in } T
\]

where \( z_0 \) is an initial state of the system, usually of the form \((\emptyset, M, f, H)\).

**Properties**

We define properties in terms of the members of a state sequence. We then say that the system has a specified property if each state of
every state sequence of the system has the property. The following notation is defined.

\[ b(S: x, y, \ldots, z) = \{0: (S, 0, x) \in b \text{ or } (S, 0, y) \in b \text{ or } \ldots \} \]

**simple-security**

A state \( v = (b, M, f, H) \) satisfies the **simple-security property** (ss-property) iff

\[ S \in S \Rightarrow [(0 \in b (S: r, w)) \Rightarrow (f_S(S) \equiv f_0(0))]. \]

It is convenient also to define:

\((S, 0, x) \in b \) satisfies the **simple security condition relative to** \( f \) (ssc rel \( f \)) iff

1. \( x = e \) or \( a \), or
2. \( x = r \) or \( w \) and \( f_S(S) \equiv f_0(0) \).

Then it is easily shown that a state \( v = (b, M, f, H) \) satisfies ss-property iff each \( (S, 0, x) \in b \) satisfies SSC rel \( f \).

***-property**

Suppose \( S' \) is a subset of \( S \). A state \( v = (b, M, f, H) \) satisfies the ***-property relative to** \( S' \) iff
\[
S \in S' \Rightarrow \begin{cases}
  (0 \in b(S:\ a)) \Rightarrow (f_0(0) \triangleright f_c(S)) \\
  (0 \in b(S:\ w)) \Rightarrow (f_0(0) = f_c(S)) \\
  (0 \in b(S:\ r)) \Rightarrow (f_c(S) \triangleright f_0(0)).
\end{cases}
\]

An immediate consequence is: if \( v \) satisfies \(*\)-property rel \( S' \) and \( S \in S' \) then

\[
[0_j \in b(S:\ a) \text{ and } 0_k \in b(S:\ r)] \Rightarrow f_0(0_j) \triangleright f_0(0_k).
\]

discretionary-security

A state \( v = (b, M, f, H) \) satisfies the discretionary-security property \((\text{ds-property})\) iff

\[
(S_i, 0_j, x) \in b \Rightarrow x \in M_{ij}.
\]

secure system

A state \( v \) is a secure state iff \( v \) satisfies the ss-property and \(*\)-property rel \( S' \) and ds-property. A state sequence \( z \) is a secure state sequence iff \( z \) is a secure state for each \( t \in T \). Call \((x, y, z) \in \Sigma(R, D, W, z_0)\) an appearance of the system. \((x, y, z) \in \Sigma(R, D, W, z_0)\) is a secure appearance iff \( z \) is a secure sequence. Finally, \( \Sigma(R, D, W, z_0) \) is a secure system iff every appearance of \( \Sigma(R, D, W, z_0) \) is a secure appearance. Similar definitions pertain for the notions.

(i) the system \( \Sigma(R, D, W, z_0) \) satisfies the ss-property,

(ii) the system satisfies \(*\)-property rel \( S' \), and

(iii) the system satisfies the ds-property.
Definition of Rule

A rule is a function \( \rho : R \times V \rightarrow D \times V \). A rule therefore associates with each request-state pair (input) a decision-state pair (output).

A rule \( \rho \) is secure-state-preserving iff \( v^* \) is a secure state whenever \( \rho (R_k, v) = (D_m, v^*) \) and \( v \) is a secure state. Similar definitions pertain for the notions

(i) \( \rho \) is ss-property-preserving,

(ii) \( \rho \) is \( * \)-property-preserving, and

(iii) \( \rho \) is ds-property-preserving.

Suppose \( \omega = \{ \rho_1, \rho_2, \ldots, \rho_s \} \) is a set of rules. The relation \( W(\omega) \) is defined by

\[
(R_k, D_m, v^*, v) \in W(\omega) \text{ iff } D_m \neq ? \text{ and } (D_m, v^*) = \rho_i (R_k, v) \text{ for a unique } i, \ 1 \leq i \leq s.
\]

Theorems

\((R_i, D_j, v^*, v) \in R \times D \times V \times V \) is an action of \( \Sigma(R, D, W, z_0) \) iff there is an appearance \((x, y, z)\) of \( \Sigma(R, D, W, z_0) \) and some \( t \in T \) such that \((R_i, D_j, v^*, v) = (x_t, y_t, z_t, z_{t-1})\).

Theorem A1:

\( \Sigma(R, D, W, z_0) \) satisfies the ss-property for any initial state \( z_0 \) which satisfies ss-property iff \( W \) satisfies the following
conditions for each action \((Rj, Dj, (b^*, M^*, f^*, H^*), (b, M, f, H)):\)

(i) each \((S, 0, x) \in b^* - b\) satisfies the simple security condition relative to \(f^*\) (SSC rel \(f^*\));

(ii) each \((S, 0, x) \in b\) which does not satisfy SSC rel \(f^*\) is not in \(b^*\).

argument:

\((\iff)\)

Suppose \(z_0 = (b, M, f, H)\) is an initial state which satisfies ss-property. Pick \((x, y, z) \in \Sigma(R, D, W, z_0)\) and write \(z_t = (b^t, M^t, f^t, H^t)\) for each \(t \in T\).

\(z_1\) satisfies ss-property.

\((x_1, y_1, z_1, z_0)\) is in \(W\). In order to show that \(z_1\) satisfies ss-property we need to show that each \((S, 0, x)\) in \(b^{(1)}\) satisfies SSC rel \(f^{(1)}\).

Notice that \(b^{(1)} = (b^{(1)} - b^{(0)}) \cup (b^{(0)} \cap b^{(1)})\) and \((b^{(1)} - b^{(0)}) \cap (b^{(1)} \cap b^{(0)}) = \emptyset\). Suppose \((S, 0, x)\) is in \(b^{(1)}\). Then either \((S, 0, x)\) is in \(b^{(1)} - b^{(0)}\) or is in \((b^{(1)} \cap b^{(0)})\).

Suppose \((S, 0, x)\) is in \((b^{(1)} - b^{(0)})\). Then \((S, 0, x)\) satisfies SSC rel \(f^{(1)}\) according to (i). Suppose \((S, 0, x)\) is in \((b^{(0)} \cap b^{(1)})\). Then \((S, 0, x)\) satisfies SSC rel \(f^{(1)}\) according to (ii). Therefore \(z_1\) satisfies ss-property.
if $z_{t-1}$ satisfies ss-property, then $z_t$ satisfies ss-property.

The argument given for "$z_1$ satisfies ss-property" applies with "t-1" substituted for "0" and "t" substituted for "1".

By induction, $z$ satisfies ss-property so that the appearance $(x, y, z)$ satisfies ss-property. $(x, y, z)$ being arbitrary, $\Sigma(R, D, W, z_0)$ satisfies the ss-property.

($\Rightarrow$)

Suppose $\Sigma(R, D, W, z_0)$ satisfies the ss-property for any initial state $z_0$ which satisfies ss-property.

Argue by contradiction. Contradiction yields the proposition

"there is some action $(x_t, y_t, z_t, z_{t-1})$ such that either

(iii) some $(S, 0, x)$ in $b(t) - b(t-1)$ does not satisfy SSC rel $f(t)$ or

(iv) some $(S, 0, x)$ in $b(t-1)$ which does not satisfy SSC rel $f(t)$ is in $b(t)$, i.e., is in $b(t-1) \cap b(t)$"

Suppose (iii). Then there is some $(S, 0, x)$ in $b(t)$ which does not satisfy SSC rel $f(t)$. Suppose (iv). Then there is some $(S, 0, x)$ in $b(t)$ which does not satisfy SSC rel $f(t)$. Therefore $z_t$ does not satisfy ss-property, $(x, y, z)$ does not satisfy ss-property, and so $\Sigma(R, D, W, z_0)$ does not satisfy ss-property, which contradicts initial assumption of the argument.
The argument is complete.

**Theorem A2:** $\Sigma(R, D, i', z_0)$ satisfies the *-property relative to $S' \subset S$ for any initial state $z_0$ which satisfies *-property relative to $S'$ iff $W$ satisfies the following conditions for each action $(R_i, D_j, (b^*, M^*, f^*, H^*), (b, M, f, H))$:

(i) for each $S \in S'$,

(a) $0 \in (b^* - b)(S:a) \Rightarrow f_0^*(0) \preceq f_c^*(S)$, and
(b) $0 \in (b^* - b)(S:w) \Rightarrow f_0^*(0) = f_c^*(S)$, and
(c) $0 \in (b^* - b)(S:r) \Rightarrow f_c^*(S) \preceq f_0^*(0)$;

(ii) for each $S \in S'$,

(a') $[0 \in b(S:a) \text{ and } f_0^*(0) \npreceq f_c^*(S)] \Rightarrow 0 \not\in b^*(S,a)$, and
(b') $[0 \in b(S:w) \text{ and } f_0^*(0) \npreceq f_c^*(S)] \Rightarrow 0 \not\in b^*(S,w)$, and
(c') $[0 \in b(S:r) \text{ and } f_c^*(S) \npreceq f_0^*(0)] \Rightarrow 0 \not\in b^*(S,r)$.

**Argument:**

($\Leftarrow$)

Suppose $z_0 = (b, M, f, H)$ is an initial state which satisfies *-property rel $S'$. Pick $(x, y, z)$ in $\Sigma(R, D, W, z_0)$ and write $z_t = (b(t), M(t), f(t), W(t))$ for each $t \in T$. 

91
$z_1$ satisfies $\ast$-property rel $S'$

$(x_1, y_1, z_1, z_0)$ is in $W$. In order to show that $z_1$ satisfies $\ast$-property rel $S'$ we need to show that:

\[
(iii) \ S \in S' \Rightarrow \begin{cases} 
0 \in b^{(1)}(S:a) \Rightarrow f^{(1)}(0) \geq f^{(1)}(S) \\
0 \in b^{(1)}(S:w) \Rightarrow f^{(1)}(0) = f^{(1)}(S) \\
0 \in b^{(1)}(S:r) \Rightarrow f^{(1)}(S) \geq f^{(1)}(0).
\end{cases}
\]

Suppose $(S, 0, x) \in b^{(1)}, \ S \in S', \ x \in \{a, w, r\}$. Then either $(S, 0, x)$ is in $(b^{(1)} - b^{(0)})$ or $(S, 0, x)$ is in $(b^{(1)} \cap b^{(0)})$.

Suppose $(S, 0, x)$ is in $(b^{(1)} - b^{(0)})$. Then (iii) is satisfied according to (i). Suppose $(S, 0, x)$ is in $(b^{(1)} \cap b^{(0)})$. Then (iii) is satisfied according to (ii). Therefore $z_1$ satisfies $\ast$-property rel $S'$.

If $z_{t-1}$ satisfies $\ast$-property rel $S'$, then $z_t$ satisfies $\ast$-property rel $S'$

The argument given for "$z_1$ satisfies $\ast$-property rel $S'$" applies with "t-1" substituted for "0" and "t" substituted for "1".

By induction, $z$ satisfies $\ast$-property rel $S'$ so that the appearance $(x, y, z)$ satisfies $\ast$-property rel $S'$. $(x, y, z)$ being arbitrary, $\Sigma(R, D, W, z_0)$ satisfies $\ast$-property relative to $S'$.

$(\Rightarrow)$

Suppose $\Sigma(R, D, W, z_0)$ satisfies $\ast$-property relative to $S'$ for any initial state $z_0$ which satisfies $\ast$-property rel $S'$. 

92
Argue by contradiction. Contradiction yields the proposition

"there is some action \((x_t, y_t, z_t, z_{t-1})\) such that either

(iv) (i) is false or
(v) (ii) is false."

Suppose (iv). Then there is some \(S \in S'\) such that (a) is false or (b) is false or (c) is false. Then \(z_t\) does not satisfy *-property rel \(S'\). Suppose (v). Then there is some \(S \in S'\) such that (a') is false or (b') is false or (c') is false. Then \(z_t\) does not satisfy *-property rel \(S'\). This leads to "\((x, y, z)\) does not satisfy *-property rel \(S'\) and so \(\Sigma(R, D, W, z_0)\) does not satisfy *-property rel \(S'\)", which contradicts initial assumption of the argument.

The argument is complete.

**Theorem A3:** \(\Sigma(R, D, W, z_0)\) satisfies the ds-property iff \(z_0\) satisfies the ds-property and \(W\) satisfies the following condition for each action \((R_i, D_j, (b^*, M^*, f^*, H^*), (b, M, f, H))\):

(i) \((S_a, 0_{a'}, x) \in b^* - b \Rightarrow x \in M^*_{a,a'}\); and

(ii) \((S_a, 0_{a'}, x) \in b \text{ and } x \notin M^*_{a,a'} \Rightarrow (S_a, 0_{a'}, x) \notin b^*\).

(\(\Leftarrow\))

If \((S_a, 0_{a'}, x) \in b^{(1)} - b^{(0)}, x \in M^{(1)}_{a,a'}\), by (i). Suppose \((S_a, 0_{a'}, x) \in b^{(1)} \cap b^{(0)}\). If \(x \notin M^{(1)}_{a,a'}\), then \((S_a, 0_{a'}, x) \notin b^{(1)}\), contrary to our supposition. Thus \(x \in M^{(1)}_{a,a'}\).
\[(S_a, 0_{a'}, x) \in b^{(1)} = (b^{(1)} - b^{(0)}) \cup (b^{(1)} \cap b^{(0)}), x \in M_{a_{a'}}^{(1)} \text{ and } z_1 \text{ satisfies the ds-property.} \]

\(\Rightarrow\)

Suppose \(\Sigma(R, D, W, z_0)\) satisfies the ds-property.

Argue by contradiction. Contradiction yields the proposition

"there is an initial state \(z_0\) satisfying the ds-property and there is some action \((x_t, y_t, z_t, z_{t-1})\) such that there is some \((S_a, 0_{a'}, x) \in b^{(t)}\) such that \(x \notin M_{a_{a'}}^{(t)}\)."

Therefore \(z_t\) does not satisfy ds-property, \((x, y, z)\) does not satisfy ds-property, and so \(\Sigma(R, D, W, z_0)\) does not satisfy ds-property, which contradicts the initial assumption of the argument.

The argument is complete.

corollary A1: \(\Sigma(R, D, W, z_0)\) is a secure system iff \(z_0\) is a secure state and \(W\) satisfies the conditions of theorems A1, A2, and A3 for each action.

theorem A4: Suppose \(\omega\) is a set of ss-property-preserving rules and \(z_0\) is an initial state which satisfies ss-property. Then \(\Sigma(R, D, W(\omega), z_0)\) satisfies ss-property.

argument

Suppose \(\Sigma(R, D, W(\omega), z_0)\) does not satisfy ss-property.
Then there is \((x, y, z)\) in \(\Sigma(R, D, W(\omega), z_0)\) which does not satisfy ss-property. Suppose \(t\) is the least element of \(T\) such that \(z_t\) does not satisfy ss-property. Since \(z_0\) satisfies ss-property, \(t > 0\). By choice of \(t\), \(z_{t-1}\) satisfies ss-property and \(z_{t-1} \neq z_t\). By definition of \(\Sigma(R, D, W(\omega), z_0)\), 
\((x_t, y_t, z_t, z_{t-1}) \in W(\omega)\). By the definition of \(W(\omega)\), there is some rule \(\rho \in \omega\) such that \(\rho(x_t, z_{t-1}) = (y_t, z_t)\). Since \(z_{t-1}\) satisfies ss-property and \(\rho(x_t, z_{t-1}) = (y_t, z_t)\) and \(\rho\) is ss-property-preserving, \(z_t\) satisfies ss-property. The contradiction shows that \(\Sigma(R, D, W(\omega), z_0)\) satisfies ss-property.

The argument is complete.

**Theorem A5:** Suppose \(\omega\) is a set of *-property preserving rules and \(z_0\) is an initial state which satisfies *-property. Then 
\(\Sigma(R, D, W(\omega), z_0)\) satisfies *-property.

**Argument:** The argument is that of theorem A4 with the substitution of *-property for ss-property.

**Theorem A6:** Suppose \(\omega\) is a set of ds-property preserving rules and \(z_0\) is an initial state which satisfies ds-property. Then 
\(\Sigma(R, D, W(\omega), z_0)\) satisfies ds-property.

**Corollary A2:** Suppose \(\omega\) is a set of secure-state-preserving rules and \(z_0\) is an initial state which is a secure state. Then 
\(\Sigma(R, D, W(\omega), z_0)\) is a secure system.

**Theorem A7:** Suppose \(v = (b, M, f, H)\) is a state which satisfies ss-property, \((S, 0, x) \neq b\), \(b^* = b \cup ((S, 0, x))\), and 
\(v^* = (b^*, M, f, H)\). Then \(v^*\) satisfies ss-property iff
(i) \( x = a \) or \( x = a \) or

(ii) \( x = r \) or \( x = w \) and \( f_s(S) \supset f_0(0) \).

**argument**

\( \Rightarrow \)

Suppose \( v^* = (b^*, M, f, H) \) satisfies ss-property. Then \( 0 \in b^* (S: r, w) \Rightarrow f_s(S) \supset f_0(0) \) by definition. Therefore (i) or (ii) holds since \( x \in \{a, w, r, a\} \).

\( \Leftarrow \)

Suppose (i). Then \( v^* \) satisfies ss-property since \( v \) does.

Suppose (ii). Then for any \( S \in S \) we have \( 0 \in b^* (S: r, w) \Rightarrow f_s(S) \supset f_0(0) \) since \( v \) satisfies ss-property. Therefore \( v^* \) satisfies ss-property.

**Theorem A8:** Suppose \( v = (b, M, f, H) \) is a state which satisfies \( *\)-property rel \( S' \subset S, S \in S', (S, 0, x) \notin b \), \( b^* = b \cup \{(S, 0, x)\} \), and \( v^* = (b^*, M, f, H) \).

\( v^* \) satisfies \( *\)-property \( ^+ \) iff

(i) if \( x = a \), then \( f_0(0) \supset f_c(S) \);

(ii) if \( x = w \), then \( f_c(S) = f_0(S) \); and

(iii) if \( x = r \), then \( f_c(S) \supset f_0(0) \).

\( ^+ \) "rel \( S' \)" is understood.
argument:

(⇒) Suppose \( v^* \) satisfies \(*\)-property. The definition of \(*\)-property applied to \( S, 0 \) and \( (S, 0, x) \) yields conditions (i), (ii), and (iii) directly.

(⇐) Suppose conditions (i) - (iii) hold. Let \( (S_i, O_j, y) \in b^* \) with \( S_i \in S' \). If \( (S_i, O_j, y) \in b \), the \(*\)-property conditions hold for \( f \) by the assumption that \( v \) satisfies \(*\)-property. If \( (S_i, O_j, y) \notin b \), \( (S_i, O_j, y) = (S, 0, x) \) and the \(*\)-property conditions hold by the initial assumption of conditions (i) - (iii). Hence \( v^* \) satisfies \(*\)-property as desired.

**Theorem A9:** Suppose \( v = (b, M, f, H) \) is a state which satisfies ds-property, \( (S_i, O_j, x) \notin b \), \( b^* = b \cup \{ (S_i, O_j, x) \} \), and \( v^* = (b^*, M, f, H) \). Then \( v^* \) satisfies ds-property iff \( x \in M_{ij} \).

argument:

(⇒) Suppose \( v^* \) satisfies ds-property. Then \( x \in M_{ij} \) by definition.

(⇐) Suppose \( x \in M_{ij} \). Then, since \( (S_i, O_j, x) \in b^* \), the proposition \( ((S_i, O_j, x) \in b^* \Rightarrow x \in M_{ij}) \) is true; therefore, \( v^* \) satisfies ds-property.

**Corollary A3:** Suppose \( v = (b, M, f, H) \) is a secure state, \( (S_i, O_j, x) \notin b \), \( b^* = b \cup \{ (S_i, O_j, x) \} \), and \( v^* = (b^*, M, f, H) \). Then \( v^* \) is a secure state iff

1. \( S_i \in S_T \) and the conditions of theorems A7 and A9 are met, or

97
(ii) $S_i \in S'$ and the conditions of theorems A7, A8, and A9 are met.

**Theorem A10**: Let $\rho$ be a rule and $\rho(\mathcal{R}_k, v) = (D_m, v^*)$, where $v = (b, M, f, H)$ and $v^* = (b^*, M^*, f^*, H^*)$.

(i) If $b^* \subseteq b$ and $f^* = f$, then $\rho$ is ss-property-preserving.

(ii) If $b^* \subseteq b$ and $f^* = f$, then $\rho$ is $*$-property-preserving.

(iii) If $b^* \subseteq b$ and $M^*_{ij} \supseteq M_{ij}$ for all $i$ and $j$, then $\rho$ is ds-property-preserving.

(iv) If $b^* \subseteq b$, $f^* = f$, and $M^*_{ij} \supseteq M_{ij}$ for all $i$ and $j$, then $\rho$ is secure-state-preserving.

**Argument:**

(i) If $v$ satisfies the ss-property, then $(S, O, x) \in b^*$ with $x = w$ or $r$ implies $(S, O, x) \in b$ so that $f_s(S) \triangleright f_o(O)$ by assumption. Hence $f_s^*(S) \triangleright f_o^*(O)$ since $f^* = f$. Thus $v^*$ satisfies ss-property and $\rho$ is ss-property-preserving.

(ii) and (iii) are proved in ways exactly analogous to the proof of (i). Implications (i), (ii), and (iii) prove implication (iv).
Rules

notation

The symbol \( \setminus \) will be used in expressions of the form \( A \setminus B \) to mean "proposition \( A \) except as modified by proposition \( B \)." Some examples follow. Suppose \( f \) is a function from the set \( \{A, B, C\} \) to the set \( \{0, 1, 3\} \) defined by:

\[
\begin{align*}
  f(A) &= 1 \text{ or } (A, 1) \in f, \\
  f(B) &= 0 \text{ or } (B, 0) \in f, \\
  f(C) &= 3 \text{ or } (C, 3) \in f.
\end{align*}
\]

Then \( f \setminus (C, 1) \) or \( f \setminus f(C) = 1 \) means

\[
\begin{align*}
  f(A) &= 1, \\
  f(B) &= 0, \\
  f(C) &= 1.
\end{align*}
\]

Suppose \( M \) is a matrix. Then \( M \setminus_{ij} a \) means the matrix obtained from \( M \) by replacing the \( (i, j) \)th element by \( a \). \( M \setminus_{ij} x \) means the matrix obtained from \( M \) by adding the element \( x \) to the \( (i, j) \)th set entry. Similarly, the notation \( f \setminus f_0 = f_0 \cup (O_{\text{NEW}(H)}, L_u) \) [see Rule 8] means the function obtained from \( f \) by replacing \( f_0 \) by \( f_0 \) plus the ordered pair \( (O_{\text{NEW}(H)}, L_u) [f_0 (O_{\text{NEW}(H)}) = L_u] \). The notation \( \text{NEW}(H) \) denotes a selection function with respect to the hierarchy \( H \) which specifies an arbitrary inactive object index.

definitions of rules

The definitions of Rules 1 to 11 are given in the following
pages. These rules preserve compatibility and assume the presence of trusted subjects.
Rule 1 (RI): get-read

Domain of RI: all \( R_k = (g, S_i, O_j, r) \) in \( R \). (Denote domain of \( R_i \) by \( \text{dom}(R_i) \).)

Semantics: Subject \( S_i \) requests access to object \( O_j \) in read-only mode \( r \).

*property function: \( *1(R_k, v) = \text{TRUE} \iff f_c(S_i) \triangleright f_o(O_j) \).

The rule:

\[
RI(R_k, v) = \begin{cases} 
(?, v) & \text{if } R_k \not\in \text{dom}(RI); \\
(yes, (b \cup (S_i, O_j, r)^{\dagger}, H, f, H)) & \text{if } (R_k \in \text{dom}(RI)) \land [r \in M_{ij}] \land [f_s(S_i) \triangleright f_o(O_j)] \land [S_i \in S_T \lor *1(R_k, v)]; \\
n(o, v) & \text{otherwise.}
\end{cases}
\]

Algorithm for RI:

\[
\text{if } R_k \not\in \text{dom}(RI) \text{ then } RI(R_k, v) = (?, v); \text{ else if } r \in M_{ij} \text{ and } [S_i \in S_T \lor *1(R_k, v)] \text{ or } [S_i \in S_T \text{ and } f_s(S_i) \triangleright f_o(O_j)] \text{ then } RI(R_k, v) = (yes, (b \cup (S_i, O_j, r), H, f, H)); \text{ else } RI(R_k, v) = (no, v);
\]

\[\dagger\text{more precisely } b \cup ((S_i, O_j, r)); \text{ braces are left out for legibility and compactness.}\]
Rule 2 (R2): get-append

Domain of R2: all \( R_k = (g, S_i, O_j, a) \in R(1) \).

Semantics: Subject \( S_i \) requests access to object \( O_j \) in append \((a)\) mode.

*-property function: \( *2(R_k, v) = TRUE \iff f_o(O_j) \preceq f_c(S_i) \).

The rule:

\[
R2(R_k, v) = \begin{cases} 
(\emptyset, v) & \text{if } R_k \notin \text{dom}(R2); \\
(yes, (b \cup (S_i, O_j, a), M, f, H)) & \text{if } [R_k \in \text{dom}(R2)] \land [a \in \gamma_{ij}] \land [S_i \in S_T \lor *2(R_k, v)]; \\
(no, v) & \text{otherwise}.
\end{cases}
\]

Algorithm for R2:

\[
\text{if } R_k \notin \text{dom}(R2) \text{ then } R2(R_k, v) = (\emptyset, v); \text{ else if } a \in \gamma_{ij} \text{ and } (S_i \in S_T \land *2(R_k, v)) \lor [S_i \in S_T] \text{ then } R2(R_k, v) = (yes, (b \cup (S_i, O_j, a), M, f, H)); \text{ else } R2(R_k, v) = (no, v);
\]
Rule 3 (R3): get-execute

Domain of R3: all $R_k = (g, S_i, O_j, e) \in R^{(1)}$.

Semantics: Subject $S_i$ requests access to object $O_j$ in execute (e) mode.

*-property function: $*3(R_k, v) = \text{TRUE}$. 

The rule:

$$R3(R_k, v) = \begin{cases} 
(?, v) & \text{if } R_k \notin \text{dom } (R3); \\
(yes, (b \cup (S_i, O_j, e), M, f, H)) & \text{if } [R_k \in \text{dom } (R3)] \& \ [e \in M_{ij}]; \\
(no, v) & \text{otherwise}.
\end{cases}$$

Algorithm for R3:

if $R_k \notin \text{dom } (R3)$ then $R3(R_k, v) = (?, v)$; else if $e \in M_{ij}$ then $R3(R_k, v) = (\text{yes}, (b \cup (S_i, O_j, e), M, f, H))$; else $R3(R_k, v) = (\text{no}, v)$;

end;
Rule 4 (R4): get-write

Domain of R4: all $R_k = (g, S_i, O_j, w) \in R^{(1)}$.

Semantics: Subject $S_i$ requests access to object $O_j$ in write ($w$) mode.

*-property function: $*4(R_k, v) = \text{TRUE} \iff f_c(S_i) = f_o(O_j)$.

The rule:

$$R4(R_k, v) = \begin{cases} (z, v) & \text{if } R_k \notin \text{dom (R4)}; \\ (\text{yes}, (b \cup (S_i, O_j, w), M,f,H)) & \text{if } [R_k \in \text{dom (R4)}] \& [w \in M_{ij}] \& [f_s(S_i) \neq f_o(O_j)] \& [S_i \in S_T \text{ or } *4(R_k, v)]; \\ \text{otherwise.} & \end{cases}$$

Algorithm for R4:

if $R_k \notin \text{dom (R4)}$ then $R4(R_k, v) = (z, v)$; else if $w \in M_{ij}$ and $[S_i \in S_T \text{ and } f_s(S_i) \neq f_o(O_j)] \text{ or } [S_i \in S^t \text{ and } *4(R_k, v)]$ then $R4(R_k, v) = (\text{yes}, (b \cup (S_i, O_j, w), M,f,H))$; else $R4(R_k, v) = (\text{no}, v)$;

end:
Rule 5 (R5): release-read/execute/write/append

Domain of R5: all $R_k = (r, S_i, o_j, x) \in R^{(1)}$, $x \in A$.

Semantics: Subject $S_i$ signals the release of access to object $o_j$ in mode $x$, where $x$ is $r$ (read-only), $e$ (execute), $w$ (write), or $a$ (append).

*-property function: $*5(R_k, v) =$ TRUE.

The rule:

$$R5(R_k, v) = \begin{cases} 
(yes, (b - (S_i, o_j, x)), M, f, H)) & \text{if } R_k \in \text{dom}(R5); \\
(?, v) & \text{otherwise.}
\end{cases}$$

Algorithm for R5:

if $R_k \notin \text{dom}(R5)$ then $R5(R_k, v) = (?, v)$;
else if $R_k \notin \text{dom}(R5)$ then $R5(R_k, v) = (yes, (b - (S_i, o_j, x)), M, f, H)$;
end;
Rule 6: give-read/execute/write/append

Domain of R₆: all \( R_k = (S_\lambda, g, S_1, O_j, x) \in R^2 \), \( x \in \{r, e, w, a\} \).

Semantics: Subject \( S_\lambda \) gives subject \( S_j \) access permission to \( O_j \) in mode \( x \), where \( x \) is \( r, e, w, \) or \( a \).

\(*\)-property function: \( *6(R_k, v) = \text{TRUE} \).

The rule:

\[
R₆(R_k, v) = \begin{cases} 
(\text{\textbar}, v) & \text{if } R_k \notin \text{dom} (R₆); \\
(\text{yes}, (b, M \setminus M_{ij} \cup \{x\}, f, H)) & \text{if } [R_i \in \text{dom} (R₆)] \land \langle [S_j \neq O_R] \& [O_s(j) \neq O_R] \& [O_s(j) \in b(S_\lambda; w)] \rangle \lor \langle [O_s(j) = O_R] \& [\text{GIVE} (S_\lambda, O_j, v)] \rangle \lor \langle [O_j = O_R] \& [\text{GIVE} (S_\lambda, O_R, v)] \rangle; \\
(no, v) & \text{otherwise}. 
\end{cases}
\]

Algorithm for R₆:

\[
\text{if } R_k \notin \text{dom} (R₆) \text{ then } R₆(R_k, v) = (\text{\textbar}, v) ; \\
\quad \text{else if } \langle [S_j \neq O_R] \& [O_s(j) \neq O_R] \& [O_s(j) \in b(S_\lambda; w)] \rangle \lor \langle [O_s(j) = O_R] \& [\text{GIVE} (S_\lambda, O_j, v)] \rangle \lor \langle [O_j = O_R] \& [\text{GIVE} (S_\lambda, O_R, v)] \rangle; \\
\quad \quad \text{then } R₆(R_k, v) = (\text{yes}, (b, M \setminus M_{ij} \cup \{x\}, f, H)); \\
\quad \text{else } R₆(R_k, v) = (\text{no}, v); \\
\text{end};
\]

\( \text{GIVE} (S_\lambda, O_k, v) = \text{TRUE} \) iff \( S_\lambda \) is allowed to give access permission to \( O_k \) in state \( v \), for \( O_k = O_R \) or \( O_s(k) = O_R \).
Rule 7 (R7): rescind-read/execute/write/append

Domain of R7: all $R_k = \{S_\lambda, r, S_i, O_j, x\} \in R(2)$, $x \in A$.

Semantics: Subject $S_\lambda$ rescinds subject $S_i$’s access permission to $O_j$ in mode $x$, where $x$ is $r$, $e$, $w$, or $a$.

*-property function: $*7(R_k, v) = TRUE$.

The rule:

$$R7(R_k, v) = \begin{cases} (\text{yes}, (b - (S_i, O_j, x), M \setminus M_{ij} - \{x\}, f, H)) & \text{if } R_k \notin \text{dom} (R7); \\ (\text{no}, v) & \text{if } [R_k \in \text{dom} (R7)] \& \left[<[O_j \neq O_R] \& [O_{s(j)} \in b(S_\lambda, w)] > \text{or} \right] \left<[O_j = O_R] \& [\text{RESCIND} (S_\lambda, O_R, v)] > \text{or} \right] \text{otherwise.} \\
\end{cases}$$

Algorithm for R7:

```plaintext
if $R_k \notin \text{dom} (R7)$ then $R7(R_k, v) = (\text{yes}, v)$; else if $<[O_j \neq O_R] \& [O_{s(j)} \in b(S_\lambda, w)] > \text{or} \left<[O_j = O_R] \& [\text{RESCIND} (S_\lambda, O_R, v)] > \text{or} \right] then $R7(R_k, v) = (\text{yes}, (b - (S_i, O_j, x), M \setminus M_{ij} - \{x\}, f, H))$; else $R7(R_k, v) = (\text{no}, v)$;

end:

RESCIND (S_\lambda, O_R, v) = TRUE iff $S_\lambda$ is allowed to rescind access permission to $O_R$ in state $v$.```
Rule 8 (Ru): create-object (preserving compatibility)

Domain of R8: all $R_k = (g, S_i, O_j, L_u) \in R^3$.

Semantics: Subject $S_i$ "generates" an object. $S_i$ requests the "creation" (i.e., attachment) of an object, denoted $O_{\text{NEW}(H)}$, having security level $L_u$, directly below $O_j$ in the hierarchy $H$ (i.e., $O_{\text{NEW}(H)} \in H(O_j)$).

*-property function: *8 $(R_k, v) = \text{TRUE}$.

The rule:

$$R8(R_k, v) = \begin{cases} (\text{yes}, (b, H, f_0 + f_o \cup O_{\text{NEW}(H)}), L_u, H \cup O_j, O_{\text{NEW}(H)})) & \text{if } R_k \notin \text{dom}(R8); \\ (\text{no}, v) & \text{if } [R_k \in \text{dom}(R8)] \& [O_j \in b(S_i; k, a)] \& [L_u \leq f_o(O_j)]; \\ & \text{otherwise.} \end{cases}$$

Algorithm for R8:

```plaintext
if $R_k \notin \text{dom}(R8)$ then $(\text{yes}, v); \text{else if } [O_j \in b(S_i; k, a)] \& [L_u \leq f_o(O_j)]$ then
  $R8(R_k, v) = (\text{yes}, (b, H, f_0 + f_o \cup O_{\text{NEW}(H)}), L_u, H \cup O_j, O_{\text{NEW}(H)}));$
else $R8(R_k, v) = (\text{no}, v);$
end;
```
Rule 9 (R9): delete-object-group

Domain of R9: all \( R_k = (S_i, O_j) \in R \)

Semantics: Subject \( S_i \) requests that object \( O_j \) be deleted (i.e., detached from the hierarchy).
This results in deletion of \( O_j \) and all objects inferior to \( O_j \) in the hierarchy.

*-property function: \(*R_k (v) = TRUE.\)

The rule:

\[
R9(R_k, v) = \begin{cases}
(\exists, v) & \text{if } R_k \notin \text{dom (R9)}; \\
(yes, (b - \text{ACCESS}(O_j), M \setminus M_{uw} + \phi; 1 \leq u \leq n, O_w \in \text{INFERIOR}(O_j), f, H - \text{SUBTREE}(O_j)) & \text{if } [R_k \in \text{dom (R9)}] \land [O_j \neq O_k] \land [O_s(j) \in b(S_i; w)]; \\
(no, v) & \text{otherwise.}
\end{cases}
\]

Algorithm for R9:

if \( R_k \notin \text{dom (R9)} \) then \( R9(R_k, v) = (\exists, v) \);
else if \([O_j \neq O_k] \land [O_s(j) \in b(S_i; w)]\) then
\[
R9(R_k, v) = \text{(yes, b - ACCESS}(O_j), M \setminus M_{uw} + \phi; 1 \leq u \leq n, O_w \in \text{INFERIOR}(O_j), f, H - \text{SUBTREE}(O_j));
\]
else \( R9(R_k, v) = (no, v) \);
end;

\( \text{INFERIOR}(O_j) = \{O_k: [O_k = O_j] \text{ or there is a set of objects } (O_1, O_2, \ldots, O_h) \text{ such that } O_k \in H(O_1), O_1 \in H(O_2), \ldots, O_k \in H(O_j)\}. \)

\( \text{SUBTREE}(O_j) = \{O_{s(k); k} = O_k \in \text{INFERIOR}(O_j))\). 

\( \text{ACCESS}(O_j) = (S \times F \\text{INFERIOR}\langle O_j \rangle \times A) \cap b. \)
Rule 10 (R10): change-subject-current-security-level

Domain of R10: all \( R_k \in (S_i, L_u) \in R^S(S). \)

Semantics: Subject \( S_i \) requests a change in its current security (value of \( f_c(S_i) \)) to \( L_u. \)

*-property function: \(*10(R_k, v) = TRUE \iff [0_j \in b (S_i : a) \Rightarrow f_0(0_j) \equiv L_u] \& [0_j \in b (S_i : y) \Rightarrow L_u = f_0(0_j)] \& [0_j \in b (S_i : x) \Rightarrow L_u \equiv f_0(0_j)] \& \)

The rule:

\[
R10(R_k, v) = \begin{cases} 
(\text{yes}, (b, M, f_c(S_i) - L_u, H)) & \text{if } R_k \in \text{dom (R10)}; \\
(\text{no, } v) & \text{if } [R_k \in \text{dom (R10)} \& [f_c(S_i) \equiv L_u] \& [S_i \in S_T \text{ or } *10(R_k, v)]; \\
& \text{otherwise.} 
\end{cases}
\]

Algorithm for R10:

if \( R_k \notin \text{dom (R10)} \) then \( R10(R_k, v) = (\text{yes}, v) \); else if \([S_i \in S_T \text{ or } *10(R_k, v)] \text{ and } [f_c(S_i) \equiv L_u] \) then \( R10(R_k, v) = (\text{yes}, (b, M, f_c(S_i) - L_u, H)) \); else \( R10(R_k, v) = (\text{no}, v) \);

end;
Rule 11 (R11): change-object-security-level

Domain of R11: all \( R_k = (r, S_i, O_j, L_u) \in R(3) \).

Semantics: Subject \( S_i \) requests that the security level of object \( O_j \) be changed (reclassified) to \( L_u \).

**-property function:** \(*_{11}(R_k, v) = \text{TRUE} \iff \) for each \( S_{\lambda} \in S' \),
\[
\begin{align*}
[(S_{\lambda}, O_j, a) \in b & \Rightarrow L_u \gg f_c(S_{\lambda})] & \\
[(S_{\lambda}, O_j, w) \in b & \Rightarrow f_c(S_{\lambda}) = L_u] & \\
[(S_{\lambda}, O_j, r) \in b & \Rightarrow f_c(S_{\lambda}) \gg L_u].
\end{align*}
\]

The rule:
\[
R11(R_k, v) = \begin{cases} 
(\text{false}, v) & \text{if } R_k \notin \text{dom (R11)}; \\
(\text{yes}, (b, M, f \setminus f_o(O_j) \cup L_u, H)) & \text{if } [R_k \in \text{dom (R11)}] \& [(S_i \in S_r \& f_c(S_i) \gg f_o(O_j)) \text{ or } f_c(S_i) \gg L_u \gg f_o(O_j)] \& [\text{for each } S \in S \{(O_j \in b(S; r, w)) \Rightarrow f_s(S) \gg L_u\}] \\
& \text{& } \& [*_{11}(R_k, v)] \& [\text{COMPAT}(v, O_j, L_u)] \& [\text{CHANGE}(v, O_j, L_u)]; \\
(\text{no}, v) & \text{otherwise.}
\end{cases}
\]

where COMPAT \((v, O_j, L_u) = \text{TRUE} \iff [L_u \gg f_o(S_i) \text{ and } f_o(O_w) \gg L_u \text{ for each } O_w \in H(O_j)], \text{ and}
\]
CHANGE\((v, O_j, L_u) = \text{TRUE} \iff [S_i \text{ is allowed to change } O_j's \text{ security level in state } v]^t
\]

\(^t\text{CHANGE is included in order to allow for additional policy enforcement for a particular system.}
Algorithm for $R_{11}$:

if $R_k \notin \text{dom}(R_{11})$ then $R_{11}(R_k, v) = (\emptyset, v)$;

else if $[S_i \in S_I \text{ or } f_c(S_i) \propto L_u \propto f_o(0)]$ and $[f_c(S_i) \propto f_o(0)]$ and

for each $S \in S$ $[0_j \in b(S: r, w) \Rightarrow (f_s(S) \propto L_u)]$ and $[*_{11}(R_k, v)]$ and

[COMPAT $(v, 0_j, L_u)$] and [CHANGE $(v, 0_j, L_u)$] then

$R_{11}(R_k, v) = (\text{yes}, (d, 0, f_o(0), L_u, H))$;

else $R_{11}(R_k, v) = (\text{no}, v)$;

end:
descriptions of rules

rule 1: get-read

Request is of the form \((g, S_i, O_j, r)\).

Subject \(S_i\) requests access to object \(O_j\) in read-only mode \((r)\).

If request is not of the proper form, then response is \(\bot\) with no state change.

Otherwise, the following conditions are checked:

(i) \(S_i\) has current access permission to \(O_j\) in read-only mode.

(ii) the security level of \(S_i\) dominates the security level of \(O_j\).

(iii) \(S_i\) is a trusted subject or the current security level of \(S_i\) dominates the security level of \(O_j\).

If conditions (i) - (iii) are met, then the response is \(\text{yes}\) and the state changes by adding an entry in the current access list indicating that \(S_i\) has read-only access to \(O_j\).

Otherwise the response is \(\text{no}\) with no state change.
rule 2: get-append

Request is of the form \((g, S_i, O_j, a)\).

Subject \(S_i\) requests access to object \(O_j\) in append mode \((a)\).

If request is not of the proper form, then response is \(\_\) with no state change.

Otherwise the following conditions are checked:

(i) \(S_i\) has current access permission to \(O_j\) in append mode.

(ii) \(S_i\) is a trusted subject or the security level of \(O_j\) dominates the current security level of \(S_i\).

If conditions (i) - (ii) are met, then the response is \(\text{yes}\) and the state changes by adding an entry to the current access list indicating that \(S_i\) has append access to \(O_j\).

Otherwise the response is \(\text{no}\) with no state change.

rule 3: get-execute

Request is of the form \((g, S_i, O_j, e)\).

Subject \(S_i\) requests access to object \(O_j\) in execute mode \((e)\).
If request is not of the proper form, then the response is ? with no state change.

Otherwise the following condition is checked:

(i) $S_i$ has current access permission to $O_j$ in execute mode.

If condition (i) is met, then the response is yes and the state changes by adding an entry to the current access list indicating that $S_i$ has execute access to $O_j$.

Otherwise the response is no with no state change.

**rule 4: get-write**

Request is of the form $(g, S_i, O_j, w)$.

Subject $S_i$ requests access to object $O_j$ in write mode ($w$).

If request is not of the proper form, then the response is ? with no state change.

Otherwise the following conditions are checked:

(i) $S_i$ has current access permission to $O_j$ in write mode.

(ii) the security level of $S_i$ dominates the security level of $O_j$. 

115
(iii) $S_i$ is a trusted subject or the current security level of $S_i$ equals the security level of $O_j$.

If conditions (i) - (iii) are met, then the response is yes and the state changes by adding an entry to the current access list indicating that $S_i$ has write access to $O_j$.

Otherwise the response is no with no state change.

rule 5: release-read/execute/write/append

Request is of the form $(r, S_i, O_j, x)$.

Subject $S_i$ signals the release of access to object $O_j$ in access mode $x$.

If request is not of the proper form, then the response is ? with no state change.

Otherwise the response is yes and the state changes by removing an entry from the current access list indicating that $S_i$ no longer has access to $O_j$ in mode $x$.

rule 6: give-read/execute/write/append

Request is of the form $(S, g, S_i, O_j, x)$. 
Subject $S_\lambda$ gives to subject $S_i$ access permission to $O_j$ in mode $x$.

If request is not of the proper form, then response is $?_\lambda$ with no state change.

Otherwise the following condition is checked:

(i) object $O_j$ is not the root object of the hierarchy and subject $S_\lambda$ has current access in write mode to $O_j$'s immediately superior object ($O_{s(j)}$) in the hierarchy

or

$O_j$ is the root object and $S_\lambda$ is allowed to give access permission to the root object in the current state.

If condition (i) is met, then the response is $\text{yes}$ and the state is changed by adding access permission for $S_i$ to $O_j$ in mode $x$ to the access permission matrix.

Otherwise the response is $\text{no}$ with no state change.

**rule 7: rescind-read/execute/write/append**

Request is of the form $(S_\lambda, r, S_i, O_j, x)$.

Subject $S_\lambda$ rescinds subject $S_i$'s access permission to $O_j$ in mode $x$. 

117
If request is not of the proper form, then response is \( ? \) with no state change.

Otherwise the following condition is checked:

(i) object \( O_j \) is not the root object of the hierarchy and subject \( S_\lambda \) has current access in write mode to \( O_j \)'s immediately superior object \( O_{s(j)} \) in the hierarchy.

or

\( O_j \) is not the root object and \( S_\lambda \) is allowed to rescind access permission to the root object in the current state.

If condition (i) is met, then response is \textit{yes} and the state changes as follows:

(i) removal of an entry from the current access list indicating that \( S_i \) no longer has access to \( O_j \) in mode \( x \).

(ii) removal of access permission for \( S_i \) to \( O_j \) in mode \( x \) from the access permission matrix.

Otherwise the response is \textit{no} with no state change.

\textbf{rule 8: create-object}

Request is of the form \((g, S_i, O_j, L_u)\).
Subject $S_i$ generates an object. $S_i$ requests creation (i.e., attachment) of an object, denoted $O_{\text{NEW}(H)}$, having security level $L_u$, directly below object $O_j$ in the hierarchy $H (O_{\text{NEW}(H)} \in H(O_j))$.

If request is not of the proper form, then response is no with no state change.

Otherwise the following conditions are checked:

(i) $S_i$ has current access to $O_j$ in write or append mode.

(ii) the security level $L_u$ dominates the security level of $O_j$.

If conditions (i) - (ii) are met, then response is yes and the state changes as follows:

(i) the security level function is updated by adding the ordered pair $(O_{\text{NEW}(H)}, L_u)$ (i.e., the security level of $O_{\text{NEW}(H)}$ is recorded as $L_u$).

(ii) the object $O_{\text{NEW}(H)}$ is added to the hierarchy such that $O_{\text{NEW}(H)}$ is directly below $O_j (O_{\text{NEW}(H)} \in H(O_j))$.

Otherwise response is no with no state change.

rule 9: delete-object-group

Request is of the form $(S_i, O_j)$. 

119
Subject $S_i$ requests that object $O_j$ be deleted (detached from the hierarchy). This results in deletion of all objects in the hierarchy which are inferior to $O_j$.

If request is not of the proper form, then response is \textit{no} with no state change.

Otherwise the following condition is checked:

(i) $S_i$ has current write access to the object immediately superior to $O_j$ ($O_{S(j)}$) and $O_j$ is not the root object.

If condition (i) is met, then response is \textit{yes} and the state changes as follows:

(i) all entries in the current access list giving subjects access to $O_j$ or any object inferior to $O_j$ in any mode are removed from the current access list.

(ii) all entries in the access permission matrix giving subjects access permission to $O_j$ or any object inferior to $O_j$ in any mode are removed from the access permission matrix.

(iii) $O_j$ and all objects inferior to $O_j$ are removed from the hierarchy.

Otherwise response is \textit{no} with no state change.
rule 10: change-subject-current-security-level

Request is of the form \((S_i, L_u)\).

Subject \(S_i\) requests that its current security level be changed to \(L_u\).

If request is not of the proper form, then response is \(?\) with no state change.

Otherwise the following conditions are checked:

(i) \(S_i\) is a trusted subject or if \(S_i\)'s security level were changed to \(L_u\), then the resulting state would satisfy \(*\)-property.

(ii) the security level of \(S_i\) dominates \(L_u\).

If conditions (i) - (ii) are met, then response is \(yes\) and the state changes by changing the current security level of \(S_i\) to \(L_u\).

Otherwise response is \(no\) with no state change.

rule 11: change-object-security-level

Request is of the form \((r, S_i, O_j, L_u)\).

Subject \(S_i\) requests that the security level of object \(O_j\) be changed to \(L_u\).

If request is not of the proper form, then response is \(?\) with no state change.
Otherwise the following conditions are checked:

(i) $S_i$ is a trusted subject and the current security level of $S_i$ dominates the security level of $O_j$;

or

the current security level of $S_i$ dominates $L_u$ and $L_u$ dominates the security level of $O_j$.

(ii) if any subject $S$ has current access to $O_j$ in read or write mode, then the current security level of $S$ dominates $L_u$.

(iii) if $O_j$'s security level were changed to $L_u$, then the resulting state would satisfy *-property.

(iv) if $O_j$'s security level were changed to $L_u$, then compatibility would be preserved in the hierarchy.

(v) $S_i$ is allowed to change $O_j$'s security level.

If conditions (i) - (v) are met, then response is yes and the state changes by changing the security level of $O_j$ to $L_u$.

Otherwise response is no with no state change.

proofs

rule 1

Suppose $v$ satisfies ss-property, *-property rel $S'$, and
ds-property and $R_k \in R$. $R_l(R_k, v) = (D_m, v^*)$ with:

(i) $v^* = v$ or

(ii) $v^* = (b \cup (S_i, 0_j, r), M, f, H)$

If (i), then $v^*$ satisfies ss-property, *-property, and ds-property since $v$ does.

Suppose (ii). If $(S_i, 0_j, r) \in b$, then $v^* = v$. Suppose $(S_i, 0_j, r) \notin b$. Then, since $f_s(S_i) \Rightarrow f_o(0_j)$ according to $R_l$, $v^*$ satisfies ss-property by theorem A7 and, since $f_c(S_i) \Rightarrow f_o(0_j)$ if $S_i \in S'$ according to $R_l$, $v^*$ satisfies *-property rel $S'$ by theorem A8 and, since $r \in M_{ij}$ according to $R_l$, $v^*$ satisfies ds-property by theorem A9.

Therefore $R_l$ is secure-state-preserving by corollary A3.

rule 2

Suppose $v$ satisfies ss-property, *-property rel $S'$, and ds-property and $R_k \in R$. $R_2(R_k, v) = (D_m, v^*)$ with

(i) $v^* = v$ or

(ii) $v^* = (b \cup (S_i, 0_j, a), M, f, H)$

Suppose (ii). If $(S_i, 0_j, a) \in b$, then $v^* = v$. Suppose $(S_i, 0_j, a) \notin b$. Then $v^*$ satisfies ss-property by theorem A7 and, since $f_o(0_j) \Rightarrow f_c(S_i)$ if $S_i \in S'$ according to $R_2$, $v^*$ satisfies *-property rel $S'$ by theorem A8 and, since $a \in M_{ij}$.
according to R2, v* satisfies ds-property by theorem A9.

Therefore R2 is secure-state-preserving by corollary A3.

rule 3

Suppose v is a secure state and R_k ∈ R.

Suppose v* = (b U (S_i, O_j, e), M, f, H) and (S_i, O_j, a) ∉ b. Then v* satisfies ss-property by theorem A7 and v* satisfies *-property rel S' by theorem A8 and, since e ∈ M_{ij} according to R3, v* satisfies ds-property by theorem A9.

Therefore R3 is secure-state-preserving by corollary A3.

rule 4

Suppose v is a secure state and R_k ∈ R.

Suppose v* = (b U (S_i, O_j, w), M, f, H) and (S_i, O_j, w) ∉ b. Then, since f_S(S_i) ∋ f_O(O_j) according to R4, v* satisfies ss-property by theorem A7 and, since f_C(S_i) = f_O(O_j) if S_i ∈ S', v* satisfies *-property rel S' by theorem A8 and, since w ∈ M_{ij} according to R4, v* satisfies ds-property by theorem A9.

Therefore R4 is secure-state-preserving by corollary A3.

rule 5

Suppose v is a secure state.
According to R5 $b^* \subseteq b$, $M^* = M$, and $f^* = f$. Therefore $v^*$ is a secure state and R5 is secure-state-preserving by theorem A10 (iv).

rule 6

Suppose $v$ is a secure state.

According to R6 $b^* = b$ and $M^* = M \cup \{x\}$. Therefore $v^*$ is a secure state and R6 is secure-state-preserving by theorem A10 (iv).

rule 7

Suppose $v$ is a secure state.

According to R7 $v^* = v$ or $v^* = (b - (S_1, O_j, x), M \setminus M_{ij} - \{x\}, f, H)$. If the latter then it is still the case that $(S_a, O_b, x) \in b \Rightarrow x \in M_{ab}$. R7 is ss-property-preserving and $^*$-property-preserving by theorem A10 (i) and (iv). Therefore $v^*$ is a secure state and R7 is secure-state-preserving.

rule 8

Suppose $v$ is a secure state.

According to R8 $b^* = b$ and $M^* = M$. Since $(S_\lambda, O_{\text{NEW}(H)}, x) \not\in b$ for any $S_\lambda$ in $S$ and $x$ in $\Lambda$, $v^*$ is a secure state and R8 is secure-state-preserving.
rule 9

Suppose $v$ is a secure state.

According to R9 if $(S_a, O_a, x) \in b^*$, then $x \in M_{aa}$, so $v^*$ is a secure state. Therefore R9 is secure-state-preserving.

rule 10

Suppose $v$ is a secure state.

According to R10 if $f^* \neq f$ then $f^* = f \setminus f_c(S_i) \leftarrow L_u$ and $R_k, v$ is true so $v^*$ is a secure state. Therefore R10 is secure-state-preserving.

rule 11

Suppose $v$ is a secure state.

According to R11 if $f^* \neq f$ then $f^* = f \setminus f_o(O_j) \leftarrow L_u$ and $R_k, v$ is true so $v^*$ is a secure state. Therefore R11 is secure-state-preserving.
REFERENCES


