

**[0114]** In another example, under the MLS policy, security levels organized under a dominance relation ( $\cong$ ), are assigned to subjects (users and their processes) and objects. The simple security property specifies that a subject is permitted read access to an object only if the subject's security level dominates the object's security level, and the \*-Property specifies that a subject is permitted write access to an object only if the object's security level dominates the subject's security level. Indirectly, the \*-Property prevents the transfer of data from an object of a higher level to an object of a lower classification. The security objective of these two rules is to prevent the direct and indirect reading of information at a level higher than the user's level.

**[0115]** FIG. 3 illustrates a configuration in system 20 that meets these security objectives in terms of permission relations and obligation relations. System 20 assumes top-secret  $\cong$  secret. The permission relations of FIG. 3 specify that users cleared to the levels of top-secret and secret are respectively assigned to the Top Secret and Secret user attributes, and objects that are classified at the top-secret and secret levels are respectively assigned to the TS and S object attributes. The S\_TS object attribute is a container for all objects classified at top-secret or secret levels. With respect to these permission relations alone, users (and their processes) that are assigned to Top Secret are only able to perform read operations on objects classified at the levels top secret and secret and users (and their processes) that are cleared secret are only able to perform read operations on objects classified at the level secret, thus showing support for the security objectives of the simple security property.

**[0116]** However, under these permission relations, a user like alice, for example, could read top secret data and subsequently write that data to a secret object. To prevent such leakage of classified information, the PM configuration of FIG. 2 comprises two event-response relations:

**[0117]** (1) read TS object  $\Rightarrow$  create deny(current process, {w}, -TS);

**[0118]** (2) read S object  $\Rightarrow$  create deny(current process, {w}, -S\_TS).

**[0119]** The first relation specifies that whenever a process successfully reads a top-secret object, it will be denied the ability to write to objects that are not in the TS container (the -symbol stands for "the complement of"). The second relation specifies that whenever a process successfully reads a secret object, it will be denied the ability to write to objects that are not in the S\_TS container (i.e., neither S nor TS).

**[0120]** In system 20, a process with its user cleared to a particular level (say top secret), can read objects at levels at or below the clearance level of the user (i.e., top secret, or secret). However, once a process has read data at a particular level (say top secret), that process can no longer write to objects below that particular level (i.e., secret).

**[0121]** Under a combination of MLS and RBAC policy instances, only processes with associated users that are Interns and Doctors and are cleared to the top secret level (e.g., alice) may perform both read and write operations on objects that are Med Records and are classified TS at the same time (e.g., mrec1).

**[0122]** Assume that user alice opens a session by authenticating herself to the system 20. As mentioned above, alice is presented with her POS, as depicted in FIG. 6. Further assume that alice issues a request to open the object mrec1 for reading and writing in a process executing on her behalf. The access control module 28 determines that mrec1 is protected under both RBAC and MLS policies, and that alice is authorized to access mrec1 in both policies, through her attributes Intern and Doctor of RBAC and Top Secret of MLS. The process

request to read mrec1 is granted by the reference mediation function. After modifying the memory image of mrec1, alice may issue a request through her process to save (write) mrec1's contents. The request issued through that process is granted based on the same considerations as before.

**[0123]** The following example illustrates how a user is prevented from leaking information from a higher security level (e.g., TS) to a lower level (e.g., S) through cut/copy and paste operation with respect to the above policy configuration. Assume the following event-response relations in addition to those defined above:

**[0124]** (3) read TS object  $\Rightarrow$  create event-response(current process create object  $\Rightarrow$  assign new object to TS);

**[0125]** (4) read S object  $\Rightarrow$  create event-response(current process create object  $\Rightarrow$  assign new object to S);

**[0126]** () copy object  $\Rightarrow$  assign clipboard(current host) to attributes(current object).

**[0127]** The effect of relation (3) is that if a process successfully reads a top-secret object, whenever that process subsequently creates an object, the new object will be assigned to the TS attribute. The effect of relation (4) is that if a process successfully reads a secret object, whenever that process subsequently creates an object, the new object will be assigned to the S attribute. The effect of relation (5) is that if a clipboard operation "copy" is performed on a source object, the object that represents the clipboard is assigned to the attributes of the source object.

**[0128]** In one example, in the case when both the copy and paste operations are performed in the same process, assume that user alice issues a request to read mrec1 (TS) in a process p. The reference mediation function grants the request. At the successful completion of the read operation, a deny relation (p, {w}, -TS) is added to the policy configuration as specified by the event-response relation (1). Also, an event-response:

**[0129]** (3.1) p creates object  $\Rightarrow$  assign new object to TS

**[0130]** is generated according to (3). Next, alice issues a request to read mrec2 (S) in the same process p. The reference mediation function again grants the request. At the successful completion of the read operation, a deny relation (p, {w}, -S\_TS) is added to the policy configuration as specified by the event-response relation (2). Also, an event-response

**[0131]** (4.1) p creates object  $\Rightarrow$  assign new object to S

**[0132]** is generated according to (4).

**[0133]** Now alice tries to copy some information from object mrec1 (TS) to object mrec2 (S) and save the latter. To copy the information from object mrec1 to the clipboard, the process p first creates an object that represents the clipboard. According to (3.1) and (4.1) the new object is assigned to both TS and S. Second, the process p actually copies the information from mrec1 to the clipboard and a "copy object" event is generated. According to relation (5), the clipboard object is assigned to all attributes of mrec1. The fact that the clipboard object is already assigned to TS does not matter. Hence, the clipboard object becomes assigned to TS, S, and Med Records.

**[0134]** Next, alice pastes the clipboard content to the mrec2 object. The paste action starts with a read operation from the clipboard object, which is classified TS and S. According to the event-response relations (1) and (2), the system 20 generates the deny relations (p, {w}, -TS) and (p, {w}, -S\_TS). The clipboard content is pasted into the mrec2 object. Finally, alice tries to save (write) the mrec2 object. Because mrec2 is not contained in TS, one of the deny relations (p, {w}, -TS) prevents the current session from saving mrec2.

**[0135]** When alice tries the reverse operation, namely to copy some information from object mrec2 (S) to object mrec1