1 Handles

A “handle” \( h \) is a random identifier, generated by the operating system. When a process creates a handle, it specifies whether it should be included in the global sets \( \tilde{A} \) and/or \( \tilde{R} \). Those handles in \( \tilde{A} \) are globally addable to labels. Those handles in \( \tilde{R} \) are globally removable from labels.

A handle \( h \) used for an integrity compartment is specified in \( \tilde{R} \) by default. A handle \( h \) used for a secrecy compartment is specified in \( \tilde{A} \) by default. A handle used for a top-secret compartment is in neither.

2 Capabilities

A handle \( h \) has associated capabilities \( a(h) \), \( r(h) \) and \( g(h) \), which denote “addition”, “removal” and “group selection”, respectively. A process that holds \( a(h) \) can freely add \( h \) to labels; a process that holds \( r(h) \) can freely remove \( h \) from labels. A process that holds \( g(h) \) also owns all capabilities in the group \( G_h \).

3 Subjects

Subjects in the system are processes. They perform actions on other subjects, and on objects.

4 Labels

A label is a set of handles. A process can add a handle \( h \) to a label \( L \) if either \( h \in \tilde{A} \) or the process owns \( a(h) \). Similarly for removal. Each process \( p \) has two labels: \( S_p \) and \( I_p \), denoting secrecy and integrity, respectively.

5 Interpretation of Labels

The label \( S_p \) describes which secrets a process has seen. For each handle \( h \in S_p \), the process is assumed to have seen all secrets associated with \( h \). The larger \( S_p \), the more types of secrets \( p \) has seen.

The label \( I_p \) describes which integrity levels \( p \) belongs to. The larger \( I_p \), the more integrity \( p \) has.
6 IPC

In general, a process $p$ can message a process $a$ if $S_p \subseteq S_q$ and $I_q \subseteq I_p$. That is, $p$ can message $q$ if $p$ has seen fewer secrets than $q$, and if $p$ is at a higher integrity level than $q$. Define a simple delta function: $\delta(p, q)$ is “true” if $p$ can send a message to $q$, and “false” otherwise.

7 Ownership

A process $p$ also has an Ownership Set $O_p$, which is a set of capabilities. It can shrink $O_p$ arbitrarily. When the OS or other processes grant $p$ capabilities, and $p$ accepts them, they appear in $O_p$.

8 Objects

An object $o$ is something that a process can act upon. Examples of objects include files on the file system, and also “Groups” as we will soon see. Objects can be shared across processes.

An object $o$ has two associated labels: $S_o$ and $I_o$, representing its secrecy and integrity labels, respectively. It also has a capability set $W_o$, that conveys which capabilities a process can present to have the privilege to write to $o$. All three (the two labels and the one capability set) are immutable.

A process $p$ can read an object $o$ if $S_o \subseteq S_p$ and if $I_p \subseteq I_o$. A process $p$ can write to an object $o$ if $S_p \subseteq S_o$, $I_o \subseteq S_p$, and $W_o \cap O_p \neq \emptyset$. That is, a process $p$ can write to object $o$ if $p$ can “send a message” to $o$, and if $p$ has at least one of the capabilities required to write to $o$.

The same delta function applies here. $\delta(o, p)$ is true iff $p$ can read $o$, and $\delta(p, o)$ is true iff $p$ can write to $o$.

9 Groups

A group is both an object and a set of capabilities. Each group $G_h$ is named uniquely by a single handle, $h$. It has labels $S_{G_h}$, $I_{G_h}$, and $W_{G_h}$ as any other object does.

A process $p$ can add a capability $c$ to $G_h$ if $c \in O_p$ and $\delta(p, G_h)$ is true. Groups are non-decreasing; no revocation is possible in the system.

If $G_h$ is a group, and the capability $g(h)$ appears in $p$’s ownership level, and $p$ can read $G_h$, then the process $p$ can claim ownership of all capabilities
in $G_h$. Of course, $G_h$ can also contain capabilities of the form $g(h')$, so groups form a directed graph.

To this effect, we define a transitive "member of" operator for groups, relative to a certain process $p$. For process $p$, capability $c$, and set $G$, we say that $c \in_p G$ if $\delta(G, p)$ is true and either $c \in G$ or there exists $h$ such that $a(h) \in G$ and $c \in_p G_h$. Note that this is a recursive definition.

The same machinery also applies to ownership labels (recall they are also capability sets), with the simplification that a process can always read its ownership label.

10 What Privileges $p$ Has

It’s useful to determine which handles a process $p$ can add or remove from labels, based on what’s in its ownership label. In particular:

$$A_p = \{ h \mid a(h) \in_p O_p \}$$
$$R_p = \{ h \mid r(h) \in_p O_p \}$$

11 Changing Labels

A process $p$ has a secrecy label $S_p$, an integrity label $I_p$, and an ownership set $O_p$. Let’s say a process wants to update a label $L$, which is either its secrecy or integrity label. A process can set a label $L$ to be $L'$ if and only if:

$$L - L' \subseteq R_p \cup \tilde{R} \quad \text{and} \quad L' - L \subseteq A_p \cup \tilde{A}$$

This is stating more formally what was described above. Handles can be removed from a label if either the process owns the removal privilege for that handle, or if the handle is universally removable. Similarly for addition.

12 Per-Channel Labels

$p$ might be a server, which can talk to many different processes. In that case, $p$ might want to act as if it has many different secrecy and integrity levels. A process $p$ can speak to $q$ with a per-channel secrecy level $S_{p \rightarrow q}$ and per-channel integrity level $I_{p \rightarrow q}$ so long as it is allowed to change its secrecy and integrity levels to $S_{p \rightarrow q}$ and $I_{p \rightarrow q}$ respectively, and it is allowed to switch back. We can apply the formulae above to see that a process can use a per-channel label $L_{p \rightarrow q}$ if:
\[(L \cup L_p \rightarrow q) - (L \cap L_p \rightarrow q) \subseteq (A_p \cup \tilde{A}) \cap (R_p \cup \tilde{R})\]

For example, say that \(S_p = \emptyset\), \(I_p = \emptyset\), \(O_p = \{a(x), r(x)\}\) and \(S_q = \{x\}\). Then \(A\) can set a per-channel secrecy label of \(S_{p \rightarrow q} = \{x\}\). This allows \(p\) and \(q\) to communicate with the same secrecy level, and therefore, to enjoy bidirectional communication.