Role-based Synchronization of Transactions to Prevent Illegal Information Flow

Tomoya Enokido
Rissho University
en0@ris.ac.jp

Valbona Barolli
Tokyo Denki University
valbona@takilab.k.dendai.ac.jp

Makoto Takizawa
Tokyo Denki University
makoto.takizawa@computer.org

Abstract
The role-based access control (RBAC) model is widely used to make systems secure. Even if every access request is authorized in the roles, illegal information flow might occur as the well-known confinement problem. In this paper, we define types of information flow relations, legal (LIF), illegal (IIF), and possibly illegal (PIF) ones $R_1 \rightarrow R_2$, $R_1 \leftarrow R_2$, and $R_1 \rightarrow R_2$ among a pair of role families $R_1$ and $R_2$, respectively. In addition, we discuss a mechanism to prevent illegal information flow where each object just keeps a role family of a transaction which has most recently manipulated the object.

1 Introduction
Information systems have to be consistent and secure in presence of various kinds of security attacks and conflicting accesses from multiple transactions to resources. In the access control models [2], only a subject $s$ is allowed to issue a method $op$ to an object $o$ only if an access right $(a, op)$ is granted to the subject $s$. In the mandatory access control model [6], only an authorizer can grant access rights to subjects. On the other hand, a subject granted an access right can grant the access right to another subject in the discretionary access control model [2]. In the role-based access control (RBAC) models [5, 8], a role is specified in a collection of access rights, which shows what subjects playing the role can do on resource objects in an enterprise. While the access control models are widely used in information systems, illegal information flow among subjects through objects may occur even if each subject can safely manipulate objects according to the access rights. This is a well-known confinement problem [3]. In the lattice-based access control (LBAC) model [6], each entity, i.e., subject and object is classified into a security class. Information flow relation among security classes is defined. Access rules on read, write, and modify are defined according to the information flow relation. However, resource objects can be only manipulated in limited ways.

In traditional database systems, the two-phase locking (2PL) protocol [1] is widely used to synchronize conflicting transactions. If a transaction $T$ holds an object, every transaction conflicting with the transaction $T$ cannot use the object, i.e., waits until the object is released by the transaction $T$. On the other hand, each transaction $T$ is assigned with timestamp $TS(T)$ showing when the transaction $T$ is initiated in the timestamp ordering (TO) scheduler [1]. If a pair of transactions $T_1$ and $T_2$ issue conflicting methods $op_1$ and $op_2$ to an object $o$, respectively, $op_1$ is performed prior to $op_2$ if $TS(T_1) < TS(T_2)$. Objects are thus manipulated by conflicting transactions in the timestamp order and no deadlock occurs. We also discuss types of role-ordering (RO) schedulers to perform a method from a transaction with a more significant role prior to another transaction with a less significant role are also discussed [4]. In the RO scheduler, general abstract types of methods are supported by each object and the discretionary model is taken. In this paper, we discuss a simpler model where every object supports only read and write methods. In addition, we consider the mandatory model.

In the role-based access control model, the transaction $T$ is assigned with a subfamily of the roles granted to the subject $s$. The subfamily of the roles is referred to as purpose of the subject $s$ to perform the transaction $T$. Let $T_1$ and $T_2$ be a pair of transactions with purposes $R_1$ and $R_2$, respectively. Suppose $T_1$ writes an object $y$ after reading an object $x$ and then $T_2$ reads the object $y$. Here, if $T_2$ is not granted an access right $(x, read)$, $T_2$ illegally reads $x$’s data from the object $y$, i.e., illegal information flow occur. In this paper, we define types of information flow relations, legal (LIF), illegal (IIF), and possibly illegal (PIF) ones $R_1 \Rightarrow R_2$, $R_1 \Leftarrow R_2$, and $R_1 \rightarrow R_2$ among a pair of role families $R_1$ and $R_2$, respectively. Then, we discuss how to schedule multiple conflicting transactions concurrently issued and how to perform conflicting methods on each object so as to prevent illegal information flow.

In section 2, we present a system model. In section 3, we define the types information flow relations among roles. In section 4, we discuss the concurrency control algorithm of multiple transactions to prevent illegal in-
formation flow and serialize conflicting methods from transactions.

2 System Model

2.1 Role-based access control (RBAC) model

Let \( S \) be a set of subjects in a system. An object is an entity which performs methods issued by subjects. Let \( O \) be a set of objects \( o_1, ..., o_m \) in a system. Each object \( o_i \) is assumed to support read and write methods for manipulating data in \( o_i \) (\( i = 1, ..., m \)). Only a subject \( s \) granted an access right \( \langle o, op \rangle \) is allowed to issue a method \( op \) to an object \( o \) in the access control models \([2, 6\text{-}8]\). In the RBAC models \([5, 8]\), a role is specified in a set of access rights. A role shows a job function in an enterprise. Let \( R \) be a set of roles \( \{r_1, ..., r_n\} \) in a system. Each role \( r_i \) is a collection \( \{a_{i1}, ..., a_{il} \} \) (\( l_i \geq 1 \)) of access rights. Let \( SR(s) \subseteq R \) be a family of roles granted to a subject \( s \). That is, the subject \( s \) plays roles \( SR(s) \). Suppose a subject \( s \) initiates a transaction \( T \). Here, \( T \) is assigned with a subfamily \( PR(T) \) of the roles granted to the subject \( s \subseteq SR(s) \). \( PR(T) \) is referred to as purpose [4] of the subject \( s \) to issue the transaction \( T \).

The access control models imply the confinement problem [3]. For example, a subject \( s_1 \) is granted a pair of access rights \( \langle f, read \rangle \) and \( \langle g, write \rangle \) on file objects \( f \) and \( g \) while another subject \( s_2 \) is only granted an access right \( \langle g, read \rangle \). Suppose the subject \( s_1 \) reads data \( x \) from the file \( f \) and then writes \( x \) to the file \( g \). Here, the subject \( s_2 \) can read the data \( x \) from the file \( g \) although \( s_2 \) is not allowed to read the data \( x \) in the file \( f \). Here, information \( x \) in the file object \( f \) illegally flows to the subject \( s_2 \) (Figure 1).

\[
\text{Figure 1. Illegal information flow.}
\]

2.2 Transactions

A write method conflicts with write and read methods while a read method conflicts with write on an object. A pair of transactions which issue conflicting methods are referred to as conflict with one another. Let \( T \) be a set \( \{T_1, ..., T_m\} \) of transactions which are concurrently performed. A schedule \( S \) of the transaction set \( T \) shows a history of methods which are issued by the transactions in the set \( T \). The precedent relation \( \rightarrow \) on transactions is defined as follows [1].

[Definition] A transaction \( T_i \) precedes a transaction \( T_j \) in a schedule \( S \) of the transaction set \( T \) (\( T_i \rightarrow T_j \)) iff (if and only if) one of the following conditions holds:

1. A method \( op_i \) issued by the transaction \( T_i \) is performed prior to a method \( op_j \) of the transaction \( T_j \) if the method \( op_i \) conflicts with the method \( op_j \).

2. \( T_i \rightarrow T_k \rightarrow T_j \) for some transaction \( T_k \) in \( T \).

A pair of transactions \( T_i \) and \( T_j \) are independent (\( T_i \parallel T_j \)) iff neither \( T_i \rightarrow T_j \) nor \( T_j \rightarrow T_i \). A schedule \( S \) is serializable iff the precedent relation \( \rightarrow \) of the transactions in the schedule \( S \) is acyclic. Conflicting transactions should be performed in a serializable schedule to keep objects consistent. In the locking protocols like two-phase locking protocols [1], transactions arbitrarily hold objects based on the first-come-first-win principal. If some transaction \( T_i \) holds an object \( o \), the other transactions which would like to use the object \( o \) have to wait for release of the object \( o \). In addition, deadlock might occur. In the timestamp ordering (TO) schedulers [1], transactions are totally ordered in their timestamps. Each transaction is assigned with timestamp showing when the transaction is initiated. Objects are manipulated by conflicting transactions in the timestamp order. That is, the earlier a transaction is, the earlier the transaction holds an object. No deadlock occurs in the TO scheduler.

3 Information Flow Relations

3.1 Transactions

We discuss what information flow to occur by performing transactions. For each role \( r \) in \( R \), let \( In(r) \) and \( Out(r) \) be input and output sets of objects \( \{o \mid \langle o, read \rangle \in r \} \) and \( \{o \mid \langle o, write \rangle \in r \} \) for a role \( r \), respectively. Let \( In(R) \) and \( Out(R) \) be \( \bigcup_{r \in R} In(r) \) and \( \bigcup_{r \in R} Out(r) \), respectively. Let \( In(T) \) and \( Out(T) \) show \( r \in R \) input and output sets of objects which a transaction \( T \) in the set \( T \) reads and writes, respectively. Here, \( In(T) \subseteq In(PR(T)) \) and \( Out(T) \subseteq Out(PR(T)) \) where \( PR(T) \) is a purpose of a transaction \( T \).

[Definition] A transaction \( T_2 \) reads from another transaction \( T_1 \) in a schedule \( S \) (\( T_1 \parallel T_2 \)) iff one of the following conditions holds:

1. \( T_1 \) precedes \( T_2 \) in \( S \) (\( T_1 \rightarrow T_2 \)) and \( Out(T_1) \cap In(T_2) \neq \emptyset \).

2. \( T_1 \parallel T_3 \parallel T_2 \) for some transaction \( T_3 \).

We define the illegal information flow to occur by performing transactions.

[Definition] A transaction \( T_2 \) illegally reads from a transaction \( T_1 \) in a schedule \( S \) (\( T_1 \leftrightarrow T_2 \)) iff the following conditions hold:

1. \( T_1 \parallel T_2 \) in \( S \).

2. \( In(T_1) - In(PR(T_2)) \neq \emptyset \).

The first condition shows that data written by a transaction \( T_1 \) might be read by another transaction \( T_2 \). The second condition means that a transaction \( T_2 \) is not granted an access right to read some object read by another transaction \( T_1 \). That is, a transaction \( T_2 \) might read data in an object on which the transaction \( T_2 \) has no read access right. In a schedule of the set \( T \), illegal information flow occurs iff there are a pair of transactions \( T_1 \) and \( T_2 \) such that the transaction \( T_2 \) illegally reads from the transaction \( T_2 \) (\( T_1 \rightarrow T_2 \)) in \( T \).
[Definition] A transaction $T_2$ legally reads from a transaction $T_1$ in schedule $S$ $(T_1 \Rightarrow T_2)$ iff the following conditions hold:
1. $T_1 \vdash T_2$.
2. $\text{In}(T_1) \subseteq \text{In}(PR(T_2))$.

[Example 1] Let us consider the following roles $ra$, $rb$, $rc$, and $rd$ on four objects $x$, $y$, $z$, and $w$:
- $ra = \{ (x, \text{read}), (y, \text{read}), (y, \text{write}), (w, \text{write}) \}$.
- $rb = \{ (x, \text{write}), (x, \text{read}), (y, \text{read}), (z, \text{read}) \}$.
- $rc = \{ (y, \text{write}), (z, \text{read}), (w, \text{write}) \}$.
- $rd = \{ (y, \text{read}), (w, \text{read}) \}$.
Here, $\text{In}(ra) = \{ x, y \}$, $\text{Out}(ra) = \{ y, w \}$, $\text{In}(rb) = \{ x, y, z \}$, $\text{Out}(rb) = \{ x \}$, $\text{In}(rc) = \{ z \}$, $\text{Out}(rc) = \{ y, w \}$, $\text{In}(rd) = \{ y, w \}$, and $\text{Out}(rd) = \emptyset$. Suppose four transactions $T_1$, $T_2$, $T_3$, and $T_4$ are initiated with roles $ra$, $rb$, $rc$, and $rd$, respectively. Let us first consider a pair of the transactions $T_1$ and $T_2$. The transaction $T_1$ can read a pair of the objects $x$ and $z$ and write a pair of the objects $y$ and $w$ since the role $ra$ is assigned to $T_1$. On the other hand, the transaction $T_2$ can read the objects $x$, $y$, and $w$ and write the object $x$. Here, suppose the transaction $T_1$ is performed prior to the other transaction $T_2$, i.e. $T_1 \prec T_2$ $(T_1 \rightarrow T_2)$ in a schedule $S$ of the transactions. That is, the transaction $T_1$ writes the object $y$ after reading the object $x$ and then the transaction $T_2$ reads the object $y$ $(T_1 \rightarrow T_2)$ as shown in Figure 2. Here, data $v$ in the object $x$ might be brought into the object $y$ by the transaction $T_1$. The data $v$ can be read by the transaction $T_2$. Since the role $rb$ includes an access right $(y, \text{read})$, the transaction $T_2$ can obtain the data in the object $y$ and can read the object $y$. Thus, no illegal information flow occur if transactions with the role $ra$ precede transactions with role $rb$. However, the transaction $T_2$ cannot read the object $y$ after performing $T_1$ because $T_2$ is not granted an access right $(x, \text{read})$ in the role $rd$. That is, $T_1 \not\rightarrow T_2$.

![Figure 2. Information flow on roles.](image)

3.2 Information Flow Relations

We define a legal information flow (LIF) relation among roles as follows:

[Definition] A role $r_1$ legally flows into a role $r_2$ $(r_1 \Rightarrow r_2)$ iff one of the following conditions holds:
1. $\text{Out}(r_1) \cap \text{In}(r_2) \neq \emptyset$ and $\text{In}(r_1) \subseteq \text{In}(r_2)$.
2. $r_1 \Rightarrow r_3 \Rightarrow r_2$ for some role $r_3$.

A role $r_1$ is legally equivalent with a role $r_2$ $(r_1 \equiv r_2)$ iff $r_1 \Rightarrow r_2$ and $r_2 \Rightarrow r_1$. Based on the legal information flow relation $\Rightarrow$ among roles, a least upper bound ( lub) $r_1 \sqcup^L r_2$ of roles $r_1$ and $r_2$ is defined to be a role $r_3$ such that $r_1 \Rightarrow r_3$ and $r_2 \Rightarrow r_3$, and there is no role $r_4$ such that $r_1 \Rightarrow r_4 \Rightarrow r_3$ and $r_2 \Rightarrow r_4 \Rightarrow r_3$. A greatest lower bound (glb) $r_1 \sqcap^L r_2$ is also defined to be a role $r_3$ such that $r_3 \Rightarrow r_1$ and $r_3 \Rightarrow r_2$, and there is no role $r_4$ such that $r_3 \Rightarrow r_4 \Rightarrow r_1$ and $r_3 \Rightarrow r_4 \Rightarrow r_2$. Thus, a lattice $(\mathcal{R}, \Rightarrow, \sqcap^L, \sqcup^L)$ is defined for a set $\mathcal{R}$ of roles and the legal information flow relation $\Rightarrow$ in a system. Here, the bottom $\sqcap^L$ is $\emptyset$ and the top $\sqcup^L$ is in a set $\mathcal{R}$ of all the roles.

[Definition] A role $r_1$ is legally independent of a role $r_2$ $(r_1 \parallel r_2)$ iff one of the following conditions holds:
1. $\text{Out}(r_1) \cap \text{In}(r_2) = \emptyset$.
2. $r_2 \not\Rightarrow r_1$.

It is noted that the legal information flow relation is transitive but the independent relation is not transitive.

In the example 1, $\text{Out}(ra) \cap \text{In}(rb) = \{ y \} \neq \emptyset$ and $\text{In}(ra) = \{ (x, y, z) \} \subseteq \text{In}(rb) = \{ (x, y, z) \}$. Hence, $ra \Rightarrow rb$, $rb \not\Rightarrow rc$ since $\text{Out}(rb) = \{ x \} \cap \text{In}(rc) = \{ z \} = \emptyset$.

The illegal information flow (IIF) relation $\not\Rightarrow$ of transactions satisfies the following property:

[Theorem] Let $T_1$ and $T_2$ be a pair of transactions with roles $r_1$ and $r_2$, respectively. “$T_1 \not\Rightarrow T_2$” does not hold if the following conditions hold:
1. $r_1 \parallel r_2$.
2. $r_1 \Rightarrow r_2$ and $T_1 \not\Rightarrow T_2$.

[Proof] If $r_1 \parallel r_2$, it is trivial that $T_1 \not\Rightarrow T_2$ holds. Next, assume $r_1 \Rightarrow r_2$ and $T_1 \not\Rightarrow T_2$. Here, $T_2$ is allowed to read every object read by $T_1$ according to the definition. Hence, $T_1 \not\Rightarrow T_2$.

This theorem means that no illegal information flow occur if a transaction with a role $r_1$ precedes every transaction with a role $r_2$ and $r_1 \Rightarrow r_2$. Next, let us consider a pair of the transactions $T_1$ and $T_4$ with the roles $ra$ and $rd$, respectively, in the example 1. If the transaction $T_1$ writes the object $w$ after reading the object $y$ and then the transaction $T_4$ reads the object $w$, there is no illegal information flow since the transaction $T_4$ is granted a read right $(y, \text{read})$ in the role $rd$ as shown in Figure 3. However, if the transaction $T_1$ writes the object $w$ after reading the object $x$, illegal information flow occur since the transaction $T_4$ is not granted a read right $(x, \text{read})$ in the role $rd$. Thus, it depends on which objects are read and written in transactions whether or not illegal information flow to occur.

![Figure 3. Information flow relations.](image)
role \( r_2 \) \((r_1 \rightarrow r_2)\) iff one of the following conditions holds:

1. \( Out(r_1) \cap In(r_2) \neq \phi \) and \( In(r_1) \cap I_n(r_2) \neq \phi \).
2. \( r_1 \rightarrow r_3 \rightarrow r_2 \) for some role \( r_3 \).

Figure 4 shows possible types of illegal information flow relations of roles \( r_1 \) and \( r_2 \).

![Image of Figure 4](image)

Figure 4. PIF relation \((r_1 \rightarrow r_2)\).

In the example 1, the role \( r_0 \) legally flows into the role \( r_d \) \((r_0 \Rightarrow r_d)\). Let \( T_1 \) and \( T_2 \) be a pair of transactions with roles \( r_1 \) and \( r_2 \), respectively. Suppose the transaction \( T_1 \) precedes the transaction \( T_2 \) \((T_1 \rightarrow T_2)\) in a schedule. If the role \( r_1 \) legally flows into the role \( r_2 \) \((r_1 \Rightarrow r_2)\), there is no illegal information flow to occur even if each transaction issues any method of the roles in any order. However, if \( r_1 \rightarrow r_2 \), illegal information flow might occur depending on which methods the transactions issue in which order.

**Definition** A role \( r_1 \) illegally flows into a role \( r_2 \) \((r_1 \leftarrow r_2)\) iff \( In(r_1) \cap In(r_2) = \phi \) and \( Out(r_1) = In(r_2) \).

![Image of Figure 5](image)

Figure 5. IIF relation.

**Definition** A role \( r_1 \) possibly illegally flows into a role \( r_2 \) \((r_1 \rightarrow r_2)\) iff one of the following conditions holds:

1. \( r_1 \rightarrow r_2 \).
2. \( In(r_1) \cap In(r_2) = \phi \) and \( r_1 \rightarrow r_3 \rightarrow r_2 \) for some role \( r_3 \).

In the example 1, let us consider a pair of the transactions \( T_3 \) and \( T_4 \) with roles \( rc \) and \( rd \), respectively. Here, \( Out(rc) = In(rd) = \{ y, w \} \). Since \( In(r_1) = \{ z \} \), \( In(r_1) \cap In(r_2) = \phi \). Here, the role \( rc \) necessarily illegally flows into the role \( rd \) \((rc \leftarrow rd)\).

**Theorem** A transaction \( T_2 \) illegally reads from another transaction \( T_1 \) in a schedule \( S \) \((T_1 \leftarrow T_2)\) iff one of the following conditions holds:

1. \( r_1 \leftarrow r_2 \) where \( PR(T_1) = \{ r_1 \} \) and \( PR(T_2) = \{ r_2 \} \).

2. \( T_1 \leftarrow T_2 \) and no transaction \( T_3 \) such that \( T_1 \leftarrow T_3 \leftarrow T_2 \) in the schedule \( S \).
3. \( In(T_1) \neq \phi \) and \( In(T_3) \neq \phi \).

**Definition** A transaction \( T_1 \) possibly illegally reads from a transaction \( T_2 \) in a schedule \( S \) \((T_1 \leftarrow T_2)\) iff one of the following conditions holds:

1. \( T_1 \leftarrow T_2 \).
2. \( T_1 \leftarrow T_3 \leftarrow T_2 \) for some transaction \( T_3 \) and \( In(T_1) \cap In(T_2) = \phi \).

The following properties hold for the information flow relations \( \Rightarrow, \rightarrow, \leftarrow, \|, \) and \( \iff \) of roles:

1. \( r_1 \Rightarrow r_3 \) if \( r_1 \Rightarrow r_2 \Rightarrow r_3 \).
2. \( r_1 \rightarrow r_2 \) if \( r_1 \rightarrow r_3 \rightarrow r_2 \).
3. \( r_1 \leftarrow r_2 \) if \( r_1 \rightarrow r_2 \Rightarrow r_3 \) or \( r_1 \Rightarrow r_2 \rightarrow r_3 \).
4. \( r_1 \rightarrow r_3 \) if \( r_1 \rightarrow r_2 \Rightarrow r_3 \).
5. \( r_1 \leftarrow r_3 \) if \( r_1 \rightarrow r_3 \).
6. \( r_1 \leftarrow r_2 \) if \( r_1 \rightarrow r_2 \).
7. \( r_1 \rightarrow r_2 \) if \( r_1 \rightarrow r_2 \).

With respect to the illegal information flow relation \( \leftarrow \) of roles, we define the lub \( r_1 \uparrow r_2 \) and glb \( r_1 \downarrow r_2 \) of a pair of roles \( r_1 \) and \( r_2 \) similarly to the lub \( \uparrow \) and glb \( \downarrow \). We also define a lattice \((R, \leftarrow, \uparrow, \downarrow)\) is defined for \( R \) set of roles and the illegal information flow relation \( \leftarrow \). The bottom \( \downarrow \) is \( \phi \) and the top \( \uparrow \) is in a set \( R \) of all the roles. A lattice \((R, \leftarrow, \uparrow, \downarrow)\) is also defined for the possibly illegal (PI) information flow relation \( \leftarrow \).

### 3.3 Information flow on role families

Let \( SR(s) \subseteq R \) show a family of roles granted to a subject \( s \). If a subject \( s \) is granted three roles \( r_1 \), \( r_2 \), and \( r_3 \), \( SR(s) = \{ r_1, r_2, r_3 \} \). A transaction \( T \) issued by a subject \( s \) is assigned a purpose \( PR(T) \subseteq SR(s) \). The subject \( s \) assigns a transaction \( T \) with a pair of the roles \( r_1 \) and \( r_2 \) in \( SR(s) \) and then issues the transaction \( T \). i.e., the purpose \( PR(T) \) is \( \{ r_1, r_2 \} \). We discuss the information flow relations among role families.

- **A role family** \( R_1 \) **legally flows** into a role family \( R_2 \) \((R_1 \rightarrow R_2)\) iff one of the following conditions holds:
  1. \( \uparrow L \in R_1 \Rightarrow \downarrow L \in R_2 r_2 \).
  2. \( R_1 \Rightarrow R_3 \Rightarrow R_2 \) for some role family \( R_3 \).

- **A role family** \( R_1 \) **is legally equivalent** with a role family \( R_2 \) with respect to the legal information flow (LII) relation \( \Rightarrow (R_1 \equiv L \Rightarrow R_2) \) iff \( R_1 \Rightarrow R_2 \) and \( R_2 \Rightarrow R_1 \).

- **A role family** \( R_1 \) **illegally flows** into a role family \( R_2 \) \((R_1 \leftarrow R_2)\) iff one of the following conditions holds:
  1. \( \uparrow P \in R_1 \leftarrow \downarrow P \in R_2 r_2 \).
  2. \( R_2 \rightarrow R_3 \rightarrow R_2 \) for some role family \( R_3 \).

- **A role family** \( R_1 \) **possibly illegally flows** into \( R_2 \) \((R_1 \leftarrow R_2)\) iff one of the following conditions holds:
  1. \( \uparrow L \in R_1 \leftarrow \downarrow L \in R_2 r_2 \).
  2. \( R_1 \leftarrow R_3 \rightarrow R_2 \) for some role family \( R_3 \).
• $R_1$ is independent of $R_2$ ($R_1 \mid\mid R_2$) iff neither $R_1$ ⇒ $R_2$, $R_2 ⇒ R_1$, $R_1 ⇔ R_2$, nor $R_2 ⇔ R_1$.

Suppose there are a pair of role families $R_1 = \{r_1, r_2\}$ and $R_2 = \{r_2, r_3\}$ as shown in Figure 6. Here, $r_1, r_2,$ and $r_3$ are roles where $r_1 = \{x, \text{read}, y, \text{write}\}$, $r_2 = \{x, \text{read}, y, \text{read}, z, \text{write}\}$, and $r_3 = \{x, \text{read}, y, z, \text{read}\}$. According to the definitions, $r_1 ⇒ r_2 ⇒ r_3$. Here, $\cup_{r \in R_1} r = r_1 \cup^{L} r_2 = r_2$ since $r_1 ⇒ r_2$ and there is no role $r'$ such that $r_1 ⇒ r' ⇒ r_2$. Similarly, $\cap_{r \in R_2} r = r_2 \cap^{L} r_3 = r_2$. Then, $R_1 ⇒ R_2$ since $\cup_{r \in R_1} r ⇒ \cap_{r \in R_2} r$.

![Figure 6. LIF relation.](image)

The following property holds for the legal information flow (LIF) relation ⇒.

**Property** Let $R_1, R_2,$ and $R_3$ be role families. $R_2 \neq R_3$ if $R_1 ⇒ R_2$ but $R_1 \neq R_3$.

**Proof** Assume $R_1 ⇒ R_2, R_2 \neq R_3$, and $R_2 ⇒ R_3$. Since $R_1 ⇒ R_2$ and $R_2 ⇒ R_3$, $R_1 ⇒ R_3$ from the property 1. This contradicts the assumption $R_1 \neq R_3$. Hence, the property holds.

Suppose a collection $R$ of roles are defined on a set $O$ of objects. First, for a pair of roles $r_1$ and $r_2$ in the set $R$, it is checked if the legal information flow (LIF) relation $r_1 ⇒ r_2$ or the illegal information flow (IIF) relation $r_1 ⇔ r_2$ holds by comparing the input and output sets according to the definitions. This is a simple computation realized in matrices $\text{LIM}$ and $\text{IIM}$ where $\text{LIM}(r_1, r_2) = 1$ if $r_1 ⇒ r_2$ else 0, and $\text{IIM}(r_1, r_2) = 1$ if $r_1 ⇔ r_2$, else 0.

Next, we have to check if $R_1 ⇒ R_2$ for a pair of role families $R_1$ and $R_2$. In order to do it, we use a basic procedure $\text{LUb}(r_1, r_2)$ and $\text{LUb}(r_1, r_2)$ to obtain least upper bounds of a pair of roles $r_1$ and $r_2$ by using the matrices $\text{LIM}$ and $\text{IIM}$, respectively. By using the lub procedures $\text{LUb}$ and $\text{LUb}$ the least upper bounds of role families $R_1$ and $R_2$, $R_1 \cup^{L} R_2$ and $R_1 \cup^{I} R_2$ are obtained by a procedure $\text{LUb}(R_1, R_2)$ and $\text{LUb}(R_1, R_2)$, respectively. If a set $T$ of transactions is a priori defined, a collection $P$ of purposes is obtained. Then, for each pair of purposes $R_1$ and $R_2$, i.e. role families, it is checked if $R_1 ⇒ R_2$. It spends computation resource to check the legal information (L) and illegal information (I) flow relations for $P$. However, the LI-relation and II-relation can be obtained at the initialization of the system. If a new type of transaction $T$ is initiated, the purpose $PR(T)$ is added to the purpose set $P$ and the LI-relation and II-relations are checked for the purpose $PR(T)$.

4. Information Flow Check

We discuss how to enhance the two-phase locking (2PL) protocol so that no illegal information occurs. A transaction $T_1$ locks an object $o$ before performing a method $op$ on the object $o$. If the object $o$ could not be locked, i.e. another transaction $T_2$ locks the object $o$, the transaction $T_1$ waits until $T_2$ unlocks the object $o$. In this paper, we assume every transaction takes the strict two phase locking protocol, i.e. a transaction $T_i$ does not issue a lock request after issuing an unlock request and every lock hold by $T_i$ is released at the end on $T_i$. Suppose the transaction $T_i$ reads an object $o_j$ in a read method $op_{ij}$ before writing the object $o_i$ in $op_{ij}$. If an access right $(o_j, \text{read})$ is not in $PR(T_i)$, illegal information flow occur. Otherwise, no illegal information flow occur. Thus, it depends on what method is performed on what object whether or not illegal information flow occur. Suppose after a write method $op_{ij}$ of a transaction $T_i$ is performed on an object $o_i$, a read method $op_{ij}$ for another transaction $T_u$ is issued to the object $o_i$. If $PR(T_i) \neq PR(T_u)$, i.e. $PR(T_i) ⇒ PR(T_u)$ or $PR(T_i) \Leftrightarrow PR(T_u)$, illegal information flow occurs. If $PR(T_1) ⇒ PR(T_u)$, illegal information flow occur. Hence, the read method $op_{ij}$ cannot be performed, i.e. $T_u$ is aborted. If $PR(T_i) \Leftrightarrow PR(T_u)$ but $PR(T_1) \Leftrightarrow PR(T_u)$, illegal information flow might occur.

Each time a method $op_{ij}$ is issued to an object $o_i$, we check if illegal information flow might occur by performing the method $op_{ij}$. A variable $o_i.P$ denotes a role family, i.e. purpose of a transaction which has most recently written the object $o_i$. $o_i.P = \phi$ in the initialization of the system. Let $op_{ij}$ be a method which is issued by a transaction $T_i$. The method $op_{ij}$ is performed on the object $o_i$ as follows [Figure 7]:

```
Perform(T_i, op_{ij}, o_i) {
    if an object $o_i$ could be locked, {
        if $op_{ij}$ = write, {
            o_i.P = PR(T_i);
            write o_i;
        }
        else { l^* op_{ij} = read */
            if $o_i.P \Leftrightarrow PR(T_i)$ or $o_i.P \Leftrightarrow PR(T_i)$,
            read o_i; l^* no illegal information flow */
            else abort T_i; }
    }
    else wait;
}
```

If $op_{ij}$ is write, a purpose $PR(T_i)$ of the transaction $T_i$ is stored in a variable $o_i.P$ and the write method $op_{ij}$ is performed on the object $o_i$. Next, suppose $op_{ij}$ is read. If $o_i.P \Rightarrow PR(T_i)$ or $o_i.P \Leftrightarrow PR(T_i)$, the transaction $T_i$ reads the object $o_i$. If $o_i.P \Leftrightarrow PR(T_i)$, the transaction $T_i$ is aborted. In this paper, if $o_i.P \Rightarrow PR(T_i)$, $T_i$ is aborted. Thus, unless $o_i.P \Rightarrow PR(T_i)$ or $o_i.P \Leftrightarrow PR(T_i)$, the transaction $T_i$ is aborted since illegal information flow might occur.
4.1 Scheduling of methods

We discuss an information flow scheduler (IFS) to prevent multiple conflicting transactions so as to keep objects consistent and prevent illegal information flow. First, suppose a pair of transactions $T_i$ and $T_u$ issue methods $op_{ti}$ and $op_{ui}$ to an object $o_i$, respectively. First, suppose every method issued by transactions is sent to a scheduler $IFS_i$ of the object $o_i$. Suppose the method $op_{ti}$ arrives prior to $op_{ui}$.

\[
\text{Schedule}(op_{ti}, op_{ui}) \{
\text{if } op_{ti} \text{ is write and } op_{ui} \text{ is read,}
\text{if } PR(T_i) \neq PR(T_u), \text{ or } PR(T_i) || PR(T_u) \{
\text{illegal information flow might occur} */
\text{cresolve}(T_i, T_u); }
\text{op_{ti} precedes op_{ui} (op_{ti} \Rightarrow op_{ui}) in } LIFS_i; }
\text{else op_{ti} and op_{ui} are arbitrarily ordered (op_{ti} \parallel op_{ui});}
\}
\]

Suppose that a transaction $T_i$ issues a method $op_{ti}$ and another transaction $T_u$ issues a method $op_{ui}$ to an object $o_i$ and $op_{ti}$ conflicts with $op_{ui}$. Here, suppose that $op_{ti}$ is write and $op_{ui}$ is read. Here, if $PR(T_u) \Rightarrow PR(T_i)$, no illegal information flow occur. However, illegal information flow might occur unless $PR(T_u) \neq PR(T_i)$, i.e., $PR(T_i)$ and $PR(T_u)$ conflict. Hence, the methods $op_{ti}$ (write) and $op_{ui}$ (read) cannot be performed in this order. Either $T_i$ or $T_u$ is aborted since illegal information flow might occur. In the procedure $cresolve(T_i, T_u)$, the conflict of the purposes $PR(T_i)$ and $PR(T_u)$ of $T_i$ and $T_u$ is resolved by aborting one of the transactions. In another way, the execution order of the methods $op_{ti}$ and $op_{ui}$ is reversed, i.e., $op_{ui}$ is performed on the object $o_i$ prior to $op_{ti}$ if $PR(T_i) \rightarrow PR(T_u)$. Here, if the transactions $T_i$ and $T_u$ issue conflicting methods $op_{ti}$ and $op_{ui}$ to another object $o_j$, the method $op_{uj}$ has to be performed prior to the method $op_{ti}$. If the method $op_{ti}$ had been already performed prior to the method $op_{ui}$, $op_{ui}$ cannot be performed prior to $op_{ti}$. Here, $T_i$ or $T_u$ has to be aborted. If the following conditions are satisfied, conflicting methods $op_{ti}$ and $op_{ui}$ are rather primarily ordered in the relation $\Rightarrow$:

**IF{ dominant conditions }**
1. Any method is performed in neither $T_i$ nor $T_u$,
2. $op_{ti} = \text{write}$ and $op_{ui} = \text{read}$,
3. $PR(T_u) \Rightarrow PR(T_i)$.

In this paper, we take the abort-oriented procedure $cresolve$ as follows:

\[
cresolve(T_i, T_u) \{
\text{abort } T_i \text{ or } T_u;
\}
\]

Since conflicting methods issued by multiple transactions are totally ordered on each object in the significance of purposes of the transactions, the transactions are serializable in the scheduler $LIFS_i$. In addition, one of transactions $T_i$ and $T_u$ is aborted if illegal information flow would occur by performing methods from the transactions $T_i$ and $T_u$ in the scheduled order.

5 Concluding Remarks

In the access control models, there might occur confinement problem, i.e., illegal information flow occur even if only authorized subjects manipulate objects in authorized ways. We take a simple model where each object supports a pair of read and write method and each subject is granted a family of roles in the mandatory access control model. In this paper, we define a legal information flow relation $\Rightarrow$ and necessarily illegal information flow relation $\rightarrow$ among role families. If $R_1 \Rightarrow R_2$ for role families $R_1$ and $R_2$, no illegal information flow occur if a transaction with purpose $R_1$ is performed prior to a transaction with $R_2$. On the other hand, if $R_1 \rightarrow R_2$, illegal information flow surely occurs. We discussed the locking protocol to keep objects secure. Then, A method is performed on an object if no illegal information flow would occur.

References


