Journal of Mobile Multimedia, Vol. 11, No. 3&4 (2015) 263–280


# A FLEXIBLE READ-WRITE ABORTION PROTOCOL TO PREVENT ILLEGAL INFORMATION FLOW AMONG OBJECTS

SHIGENARI NAKAMURA DILAWAER DUOLIKUN MAKOTO TAKIZAWA
*Department of Advanced Sciences, Hosei University*
*3-7-2, Kajino-cho, Koganei-shi, Tokyo, 184-8584, Japan*
*nakamura.shigenari@gmail.com dilewerdolkun@gmail.com makoto.takizawa@computer.org*

TOMOYA ENOKIDO
*Faculty of Business Administration, Rissho University*
*4-2-16, Osaki, Shinagawa, Tokyo, 141-8602, Japan*
*eno@ris.ac.jp*

In information systems, types of objects like multimedia objects are manipulated in various applications like mobile systems. Here, information in objects may flow to another object. Suppose a transaction reads data in an object $o_1$ and then writes data to another object $o_2$. If a transaction reads the data in the object $o_2$, the transaction can read data in the object $o_1$ even if the transaction is not granted a read access right on the object $o_1$. Here, the transaction illegally reads data in the object $o_2$. Here, information in the object $o_1$ might illegally flow to the object $o_2$. A transaction illegally writes data to an object after illegally reading data in some object. In addition, we consider a suspicious object whose data is not allowed to flow to another object. A transaction suspiciously reads data in a suspicious object. A transaction impossibly writes data to an object after reading the data in a suspicious object. Write-abortion (WA) and read-write-abortion (RWA) protocols to prevent illegal information flow are already proposed in our previous studies. In the WA protocol, a transaction is aborted once issuing an illegal or impossible write operation to an object. Read operations are meaninglessly performed since the read operations are undone due to the abortion of the transaction. In the RWA protocol, a transaction is aborted once issuing an illegal read or impossible write operation to an object. Here, read operations to be performed after an illegal read operations are lost since a transaction is aborted just on issuing an illegal read operation. In this paper, we newly propose a flexible read-write abortion (FRWA) protocol to reduce the number of meaningless and lost read operations. Here, a transaction is aborted with some probability if the transaction illegally reads data in an object. We evaluate the FRWA protocols compared with the WA and RWA protocols. We show the execution time of each transaction in the FRWA protocols is shorter than the WA protocols and more number of read operations can be performed in the RWA protocols.

*Keywords*: Illegal write; Suspicious read; Impossible write; Meaningless read; Lost read; Information flow control; Flexible read-write-abortion (FRWA) protocols;

## 1 Introduction

In information systems, types of objects, e.g. multimedia objects are manipulated by various types of applications like mobile systems. Data like traffic data obtained through sensors in mobile devices are written to databases. Thus, types of information flow from an object to another object. In the basic access control model [6], a subject $s$ like a user is first granted an access right $\langle o, op \rangle$ and then is allowed to manipulate an object $o$ in an operation $op$. In secure systems, only an authorized subject $s$ can manipulate an object $o$ in an authorized operation *op*. In the RBAC model [7, 16, 18], a role is a set of access rights. In the role-based access control (RBAC) model, a role is a set of roles. A subject is granted roles. Suppose a subject $a$ is allowed to read data in a file object $f$ and write data to a file object $g$, and another subject $b$ is not allowed to read data in the file object $g$ while allowed to read the file object $f$. Here, the subject $a$ reads data $x$ in the file $f$ and then writes the data $x$ in the file $g$. The subject $b$ can get the data $x$ from the file $g$ even if the subject $b$ is not allowed to read data in the file $f$. If the subject $b$ writes data to another file object $h$. We have to prevent illegal information flow among objects. Here, the data $x$ illegally flow from the file $f$ to the object $h$. The *legal information flow* relation ($r_i \Rightarrow r_j$) from a role $r_i$ to a role $r_j$ is defined in papers [5, 11, 13]. This means, no illegal information flow occur even if any transaction with the role $r_j$ manipulates objects after a transaction with the role $r_i$.

In order to make a system secure, information flow among objects has to be controlled [1, 2, 6, 8, 10, 18, 20]. Suppose a transaction $T_1$ with a role $r_1$ reads an object $o_1$ and writes an object $o_2$. Here, some data in the object $o_1$ might be written in the object $o_2$. Then, a transaction $T_2$ with a role $r_2$ reads the object $o_2$. By reading data in the object $o_2$, the transaction $T_2$ might get some data in the object $o_1$. If the transaction $T_2$ is not allowed to read data in the object $o_1$, the transaction $T_2$ *illegally* reads data in the object $o_2$. In addition, we consider a *suspicious* object. Here, data in a suspicious object is not allowed to flow to another object [12, 14]. A transaction *suspiciously* reads data in a suspicious object. A transaction *impossibly* writes data to an object if the transaction writes data to the object after reading data in a suspicious object.

Write-abortion (WA) [12, 14] and read-write-abortion (RWA) [15] protocols to prevent illegal information flow are discussed. In the WA protocols, a transaction is aborted only if the transaction illegally or impossibly writes an object. Read operations performed after the illegal read are *meaningless* since the transaction is aborted once a write is performed. In the RWA protocols, a transaction is aborted only if the transaction illegally reads or impossibly writes data in an object. No read operation is performed after an illegal read operation is performed even if no write operation is performed in a transaction. The read operations which cannot be performed after an illegal read operation are *lost*. In order to reduce meaningless and lost read operations, we newly propose a *flexible read-write-abortion* (*FRWA*) type of protocol in this paper. Here, a transaction is aborted if the transaction issues an illegal or impossible write operation to an object as well as the WA and RWA protocols. In addition, a transaction is aborted with some probability *ap* once issuing an illegal read operation. We evaluate the FRWA protocols compared with the WA and RWA protocols in terms of the number of transactions aborted and numbers of meaningless and lost read operations. In the FRWA protocols, more and fewer numbers of transactions are aborted than the WA protocols and RWA protocols, respectively. However, the FRWA protocols imply smaller numbers of meaningless read and lost read operations than the WA and RWA protocols, respectively. This means, the execution time of each transaction is shorter in the FRWA protocols than the WA protocols. In addition, a more number of read operations can be performed in the FRWA protocols than the RWA protocols.

In section 2, we overview related studies. In section 3, we discuss information flow relations. In section 4, we discuss the FRWA protocols to prevent illegal information flow. In section 5, we evaluate the FRWA protocols.

## 2 Related Studies

*Subjects* like users and transactions issue operations to *objects* like files and databases. Let $S$ and $O$ be sets of subjects and objects in a system, respectively. Let $OP$ be a set of operations on objects. Each object $o$ supports read ($rd$) and write ($wr$) operations. A subject issues a transaction $T$ which is a sequence of read and write operations on objects.

An access rule $\langle s, o, op \rangle$ means that a subject $s$ is allowed to manipulate an object $o$ in an operation $op$ in the basic access control (BAC) model [6]. A pair $\langle o, op \rangle$ is an *access right* or permission. A subject $s$ is allowed to manipulate an object $o$ in an operation $op$ only if the subject $s$ is granted an access right $\langle o, op \rangle$. Otherwise, the subject $s$ cannot manipulate the object $o$.

In the role-based access control (RBAC) model [7, 16, 18], each role $r$ is modeled to be a set of access rights. A subject is granted a role $r$. Each person plays a role $r$ in a society, e.g. a president role in a company and professor role in a university. Each role $r$ shows what a subject who plays the role $r$ can do in a society. Let $R$ be a set of roles in a system, $R \subseteq O \times OP$. A subject $s$ issues a transaction $T$ [9] to manipulate objects. The subject $s$ assigns the transaction a subset of roles granted to the transaction $T$. The subset is referred to as *purpose* [4, 19] of the transaction $T$.

In order to prevent illegal information flow, the lattice-based access control (LBAC) model [17] is proposed. Here, every entity, i.e. subject or object belongs to a security class. Let $SC$ be a set of security classes. An information flow relation from a security class $sc_1$ to a security class $sc_2$ ($sc_1 \rightarrow sc_2$) is defined, $\rightarrow \subseteq SC \times SC$. This means, information of a class $sc_1$ is allowed to flow to an entity of a class $sc_2$. Based on the information flow relation, access rules are defined. Suppose a subject $s$ and an object $o$ belong to security classes $sc_1$ are $sc_2$, respectively. The subject $s$ can read data in the object $o$ if $sc_2 \rightarrow sc_1$ and can write data to the object $o$ if $sc_1 \rightarrow sc_2$.

In the papers [3, 4, 5], the role-based locking (RBL) protocols and schedulers are discussed to prevent illegal information flow to occur by performing transactions in the RBAC model. The scheduler of transactions is also discussed where transactions issued by subjects with roles are ordered so that illegal information flow do not occur [3].

## 3 Information Flow Relations

### 3.1 Information flow on objects

There are two types of write operations, full write and partial write operations. In a full write operation, every attribute of an object $o$ is updated. On the other hand, only some attributes of an object $o$ are updated in a partial write. A state of an object is changed if a transaction writes data to the objects. A system state is a collection of states of objects. Thus, a system state is changed if a transaction is performed. By performing a transaction $T$ which reads data in an object $o_1$ and writes data to another object $o_2$, data in the object $o_1$ might be copied in the object $o_2$. That is, information flow occur from objects to objects by performing transactions. We consider a suspicious type of secure object [12, 14]. An object $o_i$ is *suspicious* iff any data in the object $o_i$ cannot be copied in any other object [12, 14].

**[Definition]** An object $o_i$ *flows* to an object $o_j$ ($o_i \rightarrow o_j$) in a system state if and only if (iff) one of the following conditions is satisfied:

1. A transaction writes data to an object $o_j$ after reading data in an object $o_i$.

2. For some object $o_k$, $o_i \rightarrow o_k$ and $o_k \rightarrow o_j$.

A *cone* $C(o_i)$ of an object $o_i$ is defined to be a subset $\{o_j \mid o_j \rightarrow o_i\}$ of objects. Data of an object $o_j$ in a cone $C(o_i)$ might flow to an object $o_i$. A cone $C(o_i)$ of each object $o_i$ is changed each time a transaction writes data to an object $o_i$. If some transaction fully writes data to an object $o_j$ in a cone $C(o_i)$, the object $o_j$ is not included in the cone $C(o_i)$ of every object $o_i$, i.e. $o_j \rightarrow o_i$ does not hold. Even if a transaction partially writes data to an object $o_j$, the cone $C(o_i)$ of every object $o_i$ where $o_j \rightarrow o_i$ includes the object $o_j$.

### *3.2 Information flow relations on roles*

Let $R$ be a set of roles in a system. Let $In(r_i)$ and $Out(r_i)$ be sets of objects in which data are allowed to be read and written, respectively, by a subject granted a role $r_i$, i.e. $In(r_i) = \{o \mid \langle o, rd\rangle \in r_i\}$ and $Out(r_i) = \{o \mid \langle o, wr\rangle \in r_i\}$. A pair of roles $r_i$ and $r_j$ are *equivalent* ($r_i \equiv r_j$) iff $In(r_i) = In(r_j)$ and $Out(r_i) = Out(r_j)$. A role $r_i$ *flows* to a role $r_j$ ($r_i \rightarrow r_j$) iff $Out(r_i) \cap In(r_j) \neq \phi$. A role $r_i$ is *compatible* with a role $r_j$ ($r_i \rightharpoonup r_j$) iff $r_i \not\rightarrow r_j$.

**[Definition]**

1. A role $r_i$ *legally flows* to a role $r_j$ ($r_i \Rightarrow r_j$) iff one of the following conditions holds:
   (a) $In(r_i) \neq \phi$, $r_i \rightarrow r_j$, and $In(r_i) \subseteq In(r_j)$.
   (b) For some role $r_k$, $r_i \Rightarrow r_k$ and $r_k \Rightarrow r_j$.

2. A role $r_i$ *illegally flows* to a role $r_j$ ($r_i \mapsto r_j$) iff $r_i \rightarrow r_j$ but $r_i \not\Rightarrow r_j$.

A pair of roles $r_i$ and $r_j$ are *legally equivalent* with each other ($r_i \Leftrightarrow r_j$) iff $r_i \Rightarrow r_j$ and $r_j \Rightarrow r_i$. Suppose a pair of subjects $s_i$ and $s_j$ are granted roles $r_i$ and $r_j$, respectively, and $r_i \mapsto r_j$. If the subject $s_j$ reads data in an object $o_i$ after the subject $s_i$ writes data to the object $o_i$, the subject $s_j$ might obtain data in the object $o_i$ in which the subject $s_j$ is not allowed to read data.

The *least upper bound* (*lub*) $r_i \cup r_j$ of a pair of roles $r_i$ and $r_j$ is a role $r_k$ such that $r_i \Rightarrow r_k$ and $r_j \Rightarrow r_k$ and there is no role $r_h$ such that $r_i \Rightarrow r_h \Rightarrow r_k$ and $r_j \Rightarrow r_h \Rightarrow r_k$. The *greatest lower bound* (*glb*) $r_i \cap r_j$ is a role $r_k$ such that $r_k \Rightarrow r_i$ and $r_k \Rightarrow r_j$ and there is no role $r_h$ such that $r_k \Rightarrow r_h \Rightarrow r_i$ and $r_k \Rightarrow r_h \Rightarrow r_j$. For a subset $R_i$ of the role set $R$, the least upper bound $\cup\, R_i$ is $\cup_{r_i \in R_i} r_i$. The greatest lower bound $\cap\, R_i$ is $\cap_{r_i \in R_i} r_i$. A role $r_i$ is *maximal* iff there is no role $r$ in a role subset $R_i$ such that $r_i \Rightarrow r$ and $r \not\Rightarrow r_i$. A role $r_i$ is *minimal* iff there is no role $r$ in $R_i$ such that $r \Rightarrow r_i$ and $r_i \not\Rightarrow r$. Here, $max(R_i)$ and $min(R_i)$ are subsets of maximal and minimal roles in $R_i$, respectively.

**[Definition]** [11, 13] Let $R_i$ and $R_j$ be subsets of the role set $R$ ($R_i \subseteq R$, $R_j \subseteq R$).

1. $R_i$ legally flows into $R_j$ ($R_i \Rightarrow R_j$) iff for every role $r_i$ in $R_i$ and every role $r_j$ in $R_j$, $r_i \Rightarrow r_j$ or $r_i \rightharpoonup r_j$.

2. $R_i$ illegally flows to $R_j$ ($R_i \mapsto R_j$) iff $R_i \not\Rightarrow R_j$.

It is noted $R_i \Rightarrow R_j$ iff $max(R_i) \Rightarrow min(R_j)$.

### 3.3 Illegal, suspicious, and impossible operations

We define types of read and write operations based on the information flow relations.
**[Definition]**

1. A transaction $T$ *illegally reads* data in an object $o_i$ iff there is some object $o_j$ in the cone $C(o_i)$ whose access right $\langle o_j, rd\rangle$ is not granted to the transaction $T$.
2. A transaction $T$ *suspiciously reads* data in an object $o_i$ iff $o_i$ is suspicious or there is some suspicious object $o_j$ in the cone $C(o_i)$.
3. A transaction $T$ *illegally writes* data to an object $o_i$ iff the transaction $T$ writes data to the object $o_i$ after illegally reading data in some object.
4. A transaction $T$ *impossibly writes* data to an object $o_i$ iff the transaction $T$ writes data to the object $o_i$ after suspiciously reading data in some object.

We consider a transaction $T$ writes data to an object $o_j$ after reading data in another object $o_i$. Suppose the object $o_i$ is suspicious. Here, the transaction $T$ impossibly writes data to the object $o_j$ since the transaction $T$ suspiciously reads data in the object $o_i$. Next, the transaction $T$ is not granted a read access right $\langle o_k, rd\rangle$ on some object $o_k$. Suppose some transaction writes data to an object $o_i$ after reading data in an object $o_k$. Here, the object $o_k$ is included in a cone $C(o_i)$. Here, the transaction $T$ illegally reads data in the object $o_i$ and then illegally writes data to the object $o_j$.

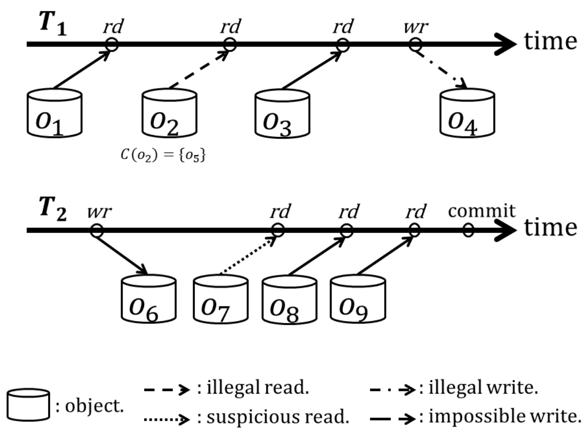


Fig. 1. Information flow on objects.

A transaction $T$ is a sequence of read and write operations on objects. An operation $op_1$ *precedes* another operation $op_2$ ($op_1 \rightarrow_T op_2$) iff $op_1$ is performed before $op_2$ in a transaction $T$. The precedent relation $\rightarrow_T$ shows an execution sequence of operations in a transaction $T$.
**[Definition]**

1. A read operation $rd$ is *meaningless* in a transaction $T$ iff $ir \rightarrow_T rd$ for some illegal or suspicious read operation $ir$ and there is some write operation $wr$ such that $rd \rightarrow_T wr$.

2. A read operation $rd$ is performed after an illegal or suspicious read operation $ir$ and some write operation $wr$ is performed after the read operation $rd$.
3. A read operation $rd$ is performed after an illegal or suspicious read operation $ir$ and no write operation is performed after the read operation $rd$.
4. A read operation $rd$ is *lost* in a transaction $T$ iff $ir \rightarrow_T rd$ for some illegal or suspicious read operation $ir$ and there are no write operation $wr_1$ and $wr_2$ such that $ir \rightarrow_T wr_1 \rightarrow_T rd$ and $rd \rightarrow_T wr_2$.

A read operation $rd$ is *safe* iff there is no write operation $wr$ such that $rd \rightarrow_T wr$ in a transaction $T$.

Suppose there are nine objects $o_1, \ldots, o_9$ and a pair of transactions $T_1$ and $T_2$ as shown in Figure 1. Here, the cone $C(o_2)$ of an object $o_2$ is assumed to include an object $o_5$. We also assume the transaction $T_1$ is not allowed to read data in the object $o_5$. We assume an object $o_7$ is suspicious. The transaction $T_1$ first reads data in the object $o_1$ and then reads data in the object $o_2$. Here, the transaction $T_1$ illegally reads data in the object $o_2$ since the transaction $T_1$ is not allowed to read data in the object $o_5$ in the cone $C(o_2)$. After that, the transaction $T_1$ reads data in an object $o_3$. Then, the transaction $T_1$ illegally writes data to an object $o_4$ since the transaction $T_1$ has illegally read data in the object $o_2$. The read operation on the object $o_3$ is meaningless.

Next, the transaction $T_2$ writes data to the object $o_6$ and suspiciously reads data in the object $o_7$. Then, the transaction $T_2$ reads data in the objects $o_8$ and $o_9$, and commits. The read operations on the objects $o_8$ and $o_9$ are lost. Every read operation is safe since no write operation is performed after the read operation.

## 4 Synchronization Protocols

### *4.1 Types of synchronization protocols*

In order to prevent illegal information flow among objects, types of synchronization protocols are proposed, the write abortion (WA) [12, 14] and read-write abortion (RWA) [15] synchronization protocols. In the WA protocol, a transaction is aborted once issuing an illegal write operation to an object.

**[WA protocol]**

Read: A transaction $T$ reads data in an object $o_i$.

Write: If a transaction $T$ issues an illegal or impossible write operation to an object $o_i$, the transaction $T$ is aborted. Otherwise, the transaction $T$ writes data to the object $o_i$

Figure 2 shows a state transition diagram of the WA protocols. Here, $rd$, $wr$, $ir$, and $sr$ show events that read, write illegal read, and suspicious read operations are issued, respectively. In the WA protocols, a transaction is not aborted even if the transaction issues an illegal read operation to an object. A transaction is aborted if the transaction issues an illegal or impossible write operation to an object.

In the RWA protocols, a transaction is aborted only if the transaction issues an illegal read or impossible write operation to an object.

**[RWA protocol]**

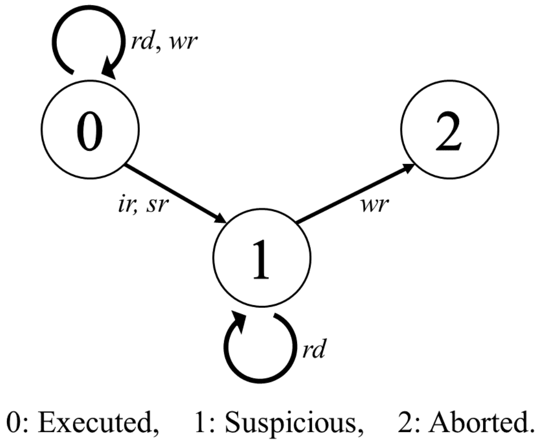


Fig. 2. State transition diagram of the WA protocols.

Read: If a transaction $T$ issues an illegal read operation to an object $o_i$, the transaction $T$ is aborted. Otherwise, the transaction $T$ reads data in the object $o_i$.

Write: If a transaction $T$ issues an impossible write operation to an object $o_i$, the transaction $T$ is aborted. Otherwise, the transaction $T$ writes data to the object $o_i$.

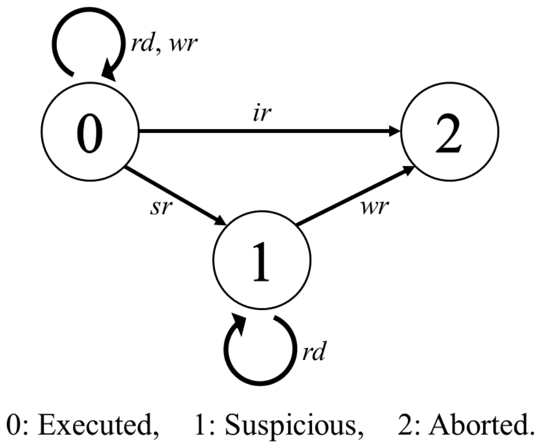


Fig. 3. State transition diagram of the RWA protocols.

Figure 3 shows a state transition diagram of the RWA protocols. A transaction is aborted once issuing an illegal read operation to an object. A transaction is aborted once issuing an write operation to an object after suspiciously reading data in a suspicious object.

In Figure 4, a pair of transactions $T_1$ and $T_2$ are performed in the WA and RWA protocols, respectively. Suppose the read operation $rd_2$ is illegal in the WA and RWA protocols. In the

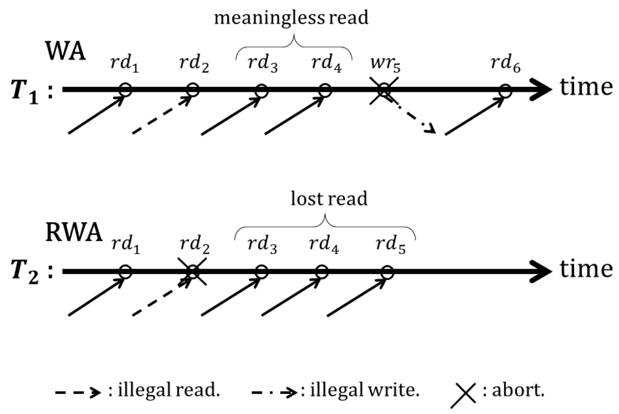


Fig. 4. Meaningless and lost read operations.

WA protocols, the read operations $rd_3$ and $rd_4$ are meaningless since the transaction is aborted when the transaction $T_1$ issues the illegal write operation $wr_5$. In the RWA protocols, the transaction $T_2$ is aborted on issuing the illegal read operation $rd_2$. Here, the read operations $rd_3$, $rd_4$, and $rd_5$ are lost. Since the transaction $T_2$ issues no write operation, no illegal information flow occur even if the read operations $rd_2$, $rd_3$, $rd_4$, and $rd_5$ are performed. As shown in this example, meaningless read operations are performed in the WA protocols and lost read operations are not performed in the RWA protocols. In the WA protocols, read-oriented operations are efficiently performed since read transactions are not aborted. However, meaningless read operations are performed. In the RWA protocols, write-oriented operations are efficiently performed since write transactions are not aborted. However, read operations are lost, i.e. not performed.

We newly propose a *flexible read-write-abortion* (*FRWA*) type of protocol in order to reduce the number of meaningless and lost read operations in this paper.

**[FRWA protocol]**

Read: If a transaction $T$ issues an illegal read operation to an object $o_i$, the transaction $T$ is aborted with probability *ap*. Otherwise, the transaction $T$ reads data in the object $o_i$.

Write: If a transaction $T$ issues an illegal or impossible write operation to an object $o_i$, the transaction $T$ is aborted. Otherwise, the transaction $T$ writes data to the object $o_i$.

Figure 5 shows a state transition diagram of the FRWA protocols, where *ap* shows the abortion probability. In the FRWA protocols, a transaction is aborted with abortion probability *ap* if the transaction issues an illegal read operation to an object. In addition, a transaction is aborted if the transaction issues an illegal or impossible write operation to an object. If $ap = 0$, the FRWA protocol is the same as the WA protocol. If $ap = 1$, the FRWA protocol is the same as the RWA protocol.

Table 1 summarizes the properties of WA, RWA, and FRWA protocols.

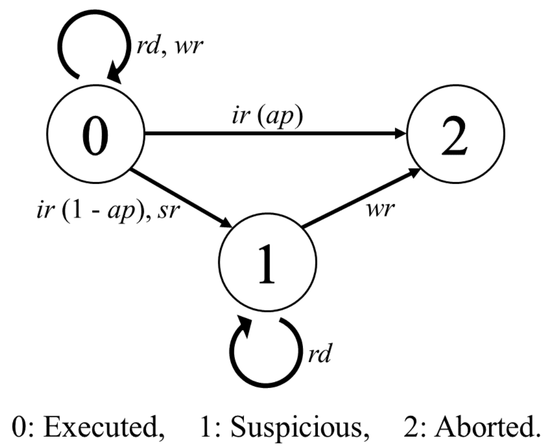


Fig. 5. State transition diagram of the FRWA protocols.

Table 1. Summary of the WA, RWA, and FRWA protocols.

| | illegal read | suspicious read | illegal write | impossible write |
|---|---|---|---|---|
| WA | ○ | ○ | × | × |
| RWA | × | ○ | − | × |
| FRWA | △ | ○ | × | × |

× : abort, ○ : non-abort, △ : abort or non-abort, − : not occur.

There are two types of synchronization protocols, RBS (role-based synchronization) and OBS (object-based synchronization) on how to check if a read operation is illegal.

In the RBS protocol, a transaction $T$ and an object $o_i$ hold sets $T.R$ and $o_i.R$ of roles, respectively [11, 13]. The role sets $T.R$ and $o_i.R$ are manipulated as follows:

1. Initially, $o_i.R = \phi$ for every object $o_i$; Initially, $T.R = T.P$ for every transaction $T$;
2. If a transaction $T$ writes data to an object $o_i$, $o_i.R = o_i.R \cup T.R$;
3. If a transaction $T$ reads data in an object $o_i$, $T.R = T.R \cup o_i.R$;

A transaction $T$ is referred to as *hold* a role $r$ if $r \in T.R$. An object $o_i$ *holds* a role $r$ if $r \in o_i.R$.

Suppose a transaction $T$ issues a read operation to an object $o_i$. It is checked if the read operation is illegal by using the role sets $o_i.R$ and $T.R$. If $o_i.R \mapsto T.R$, the transaction $T$ illegally reads data in the object $o_i$ as discussed.

In the OBS protocol, a transaction $T$ and an object $o_i$ hold sets $T.O$, $T.C$, and $o_i.C$ of objects, respectively [12, 14]. The object sets $T.O$, $T.C$, and $o_i.C$ are manipulated as follows:

1. Initially, $o_i.C = \phi$ for every object $o_i$; Initially, $T.O = T.C = \phi$ for every transaction $T$;
2. If a transaction $T$ fully writes data to an object $o_i$, $o_i.C = T.C \cup T.O$;
3. If a transaction $T$ partially writes data to an object $o_i$, $o_i.C = o_i.C \cup T.C \cup T.O$;
4. If a transaction $T$ reads data in an object $o_i$, $T.C = T.C \cup o_i.C$ and $T.O = T.O \cup \{o_i\}$;

A transaction $T$ is referred to as *hold* an object $o_j$ if $o_j \in T.C$. An object $o_i$ *holds* an object $o_j$ if $o_j \in o_j.C$.

Suppose a transaction $T$ issues a read to an object $o_i$. It is checked if the read operation is illegal by using the object and role sets $o_i.C$. and $T.P$. If $o_i.C \mapsto In(T.P)$, the transaction $T$ illegally reads data in the object $o_i$ as discussed.

### *4.2 Implementation*

We discuss how to implement the WA, RWA, and FRWA protocols. In the protocols, the following variables are manipulated for each transaction $T$ and each object $o_i$:

1. $T.P$ = purpose of the transaction $T$, i.e. collection of roles assigned to the transaction $T$.
2. $T.R$ = set of roles, initially $\phi$.
3. $T.O$ = set of objects in which the transaction $T$ reads data.
4. $T.C = \cup_{o_j \in T.O} C(o_j)$, set of objects in cones of objects which the transaction $T$ reads.
5. $o_i.R$ = set of roles, initially $\phi$.
6. $o_i.C$ = cone $C(o_i)$ of the object $o_i$ which is a set of objects, initially $\phi$.

Each variable is implemented in a bitmap. In the variables $T.O$, $T.C$, and $o_i.C$, the $i$th bit shows an object $o_i$ ($i = 1, \ldots, n$). If an object $o_i$ is included in the variable, the $i$th bit is 1 else 0. In the variables $T.P$, $T.R$, and $o_i.R$, the $j$th bit shows a role $r_j$ ($j = 1, \ldots, rn$). If a role $r_j$ is in the variable, the $j$th bit is 1 else 0. For a pair of bitmaps $B_1$ and $B_2$, $B_1 \cup B_2$ shows a disjunction of $B_1$ and $B_2$.

First, we would like to present how a transaction $T$ reads and writes data in an object $o_i$. We consider how to implement a write operation. As presented before, there are a pair of full write (fwrite) and partial write (pwrite) operations. write ($o_i$) shows a basic write operation on an object $o_i$.

**[RBS_fwrite ($T$, $o_i$)]** $o_i.R = T.R$; write ($o_i$);
**[RBS_pwrite ($T$, $o_i$)]** $o_i.R = o_i.R \cup T.P$; write ($o_i$);
**[OBS_fwrite ($T$, $o_i$)]** $o_i.C = T.C$; write ($o_i$);
**[OBS_pwrite ($T$, $o_i$)]** $o_i.C = o_i.C \cup T.C$; write ($o_i$);

In the RBS type of protocol, if a transaction $T$ writes data to an object $o_i$, the roles of the transaction $T$ are recorded in the variable $R$ of the object $o_i$. In the OBS type of protocol, objects in which the transaction $T$ reads data are recorded in the variables $T.C$ and $T.O$ of the transaction $T$. In addition, the cones of the objects are recorded in the variable $T.C$. If the object $o_i$ is fully written, the variables $o_i.R$ and $o_i.C$ are replaced with the variables $T.R$ and $T.C$, respectively. If the object $o_i$ is partially written, roles and objects recorded in the variables $o_i.R$ and $o_i.C$ are added to the variables $T.R$ and $T.C$, respectively.

Next, we consider a read operation on an object $o_i$. Here, read ($o_i$) shows a basic read operation on an object $o_i$.

**[RBS_read ($T$, $o_i$)]** $T.R = T.R \cup o_i.R$; read ($o_i.$);
**[OBS_read ($T$, $o_i$)]** $T.O = T.O \cup \{o_i\}$; $T.C = T.C \cup \{o_i\} \cup o_i.C$; read ($o_i$);

In the RBS type of protocol, roles $o_i.R$ recorded in the object $o_i$ are recorded in the variable $T.R$ of the transaction $T$. In the OBS type of protocol, objects $o_i.C$ recorded in the object $o_i$, i.e. the cone $C(o_i)$ are recorded in the variable $T.C$ of the transaction $T$.

A transaction $T$ checks if a read operation is illegal and a write operation is impossible as follows:

**[RBS_ilread ($T$, $o_i$)]** If $o_i.R \mapsto T.R$, true;
**[OBS_ilread ($T$, $o_i$)]** If $o_i \notin In(T.P)$ for some object $o_i$ in $o_i.C$, true.

In an RBS type of protocol, the information flow relation on the roles in the variables $o_i.R$ and $T.P$ are checked. If information flow from the object $o_i$ to the transaction $T$ is illegal, i.e. $o_i.R \mapsto T.R$, the read operation is illegal. By analyzing every role, an illegal flow relation $r_i \mapsto r_j$ is found for some pair of roles $r_i$ and $r_j$. The illegal flow relation is realized in a matrix LIF where LIF$[i, j] = 1$ if $r_i \not\to r_j$, else 0.

In an OBS type of protocol, every object in the cone $C(o_i)$ of the object $o_i$ is checked. If a transaction is not allowed to read data in some object in the cone $C(o_i)$, the read operation on the object $o_i$ is illegal.

A transaction $T$ checks if a write operation on an object $o_i$ is impossible or suspicious by the following functions:

**[impwrite ($T$, $o_i$)]** If an object $o_j$ in $T.C$ is suspicious, true;
**[spread ($T$, $o_i$)]** If an object $o_i$ is suspicious, a transaction $T$ is marked *suspicious.*

The WA and RWA protocols are implemented as follows;

**[WA-RBS protocol]**

Read: **if** RBS_ilread ($T$, $o_i$), mark ($T$);
RBS_read ($T$, $o_i$);

Write: **if** $T$ is marked *illegal* or impwrite ($T$, $o_i$), abort ($T$);
RBS_write ($T$, $o_i$);

**[WA-OBS protocol]**

Read: **if** OBS_ilread ($T$, $o_i$), mark ($T$); **else** OBS_read ($T$, $o_i$);

Write: **if** $T$ is marked *illegal* or impwrite ($T$, $o_i$), abort ($T$);
OBS_write ($T$, $o_i$);

**[RWA-RBS protocol]**

Read: **if** RBS_ilread ($T$, $o_i$), abort ($T$); **else** RBS_read ($T$, $o_i$);

Write: **if** impwrite ($T$, $o_i$), abort ($T$); **else** RBS_write ($T$, $o_i$);

**[RWA-OBS protocol]**

Read: **if** OBS_ilread ($T$, $o_i$), abort ($T$); **else** OBS_read ($T$, $o_i$);

Write: **if** impwrite ($T$, $o_i$), abort ($T$); **else** OBS_write ($T$, $o_i$);

The procedure $AP(T)$ randomly takes true or false with abortion probability $ap$ for a transaction $T$. The FRWA-RBS and FRWA-OBS protocols are implemented as follows;

**[FRWA-RBS protocol]**

Read: **if** RBS_ilread ($T$, $o_i$),
{if $AP(T)$ is satisfied, abort ($T$) **else** mark($T$);}
**else if** RBS_spread ($T$, $o_i$), mark ($T$);
RBS_read ($T$, $o_i$);

Write: **if** $T$ is marked, abort ($T$);
**else** RBS_write ($T$, $o_i$);

**[FRWA-OBS protocol]**

Read: **if** OBS_spread ($T$, $o_i$),
{if $AP(T)$ is satisfied, abort ($T$) **else** mark(T);}
**else if** OBS_spread ($T$, $o_i$), mark ($T$);
OBS_read ($T$, $o_i$);

Write: **if** $T$ is marked, abort ($T$);
**else** OBS_write ($T$, $o_i$);

## 5 Evaluation

### *5.1 Environment*

As discussed in this paper, there occur no illegal information flow in the WA, RWA, and FRWA protocols while some transactions are aborted, meaningless read operations are performed, and lost read operations are not performed. We evaluate the FRWA-RBS and FRWA-OBS protocols in terms of number of transactions aborted compared with the WA, RWA, and NBS protocols on an object set $O$ and a role set $R$. The NBS protocol means a protocol where no information flow control is implemented. Here, each transaction just reads data in and writes data to objects and no transaction is aborted. In the WA protocols, meaningless read operations which are not necessarily to be performed are performed. The more number of meaningless read operations, the longer it takes to perform transactions. In the RWA protocols, lost read operations which can be performed but can not be performed once an illegal read operation is issued.

In the evaluation, there are $n$ objects $o_1 \ldots o_n$, $O = \{o_1, \ldots, o_n\}$. Each object $o_i$ supports *read* ($rd$) and *write* ($wr$) operations. Let $\mu$ be a ratio of the number of suspicious objects out of $n$ objects ($0 \leq \mu \leq 1$). There is no suspicious object if the suspicious ratio $\mu$ is 0. A number $rn$ ($\geq 1$) of roles $r_1, \ldots, r_{rn}$ are defined by randomly selecting access rights on $n$ objects $o_1, \ldots, o_n$. $R = \{r_1, \ldots, r_{rn}\}$. Here, $mran$ ($\leq 2n$) shows the maximum number of access rights to be included in each role $r_i$. Each role $r_i$ is composed of $an_i$ ($1 \leq an_i \leq mran$) access rights. In the evaluation, the number $an_i$ for each role $r_i$ is randomly selected out of numbers 1, 2, …, $mran$. Then, the number $an_i$ of access rights are randomly selected in $2n$ access rights $\langle o_1, rd\rangle$, $\langle o_1, wr\rangle$, …, $\langle o_n, rd\rangle$, $\langle o_n, wr\rangle$ so that no duplicate access right is included in each role $r_i$ ($i = 1, \ldots, rn$).

There are $tn$ ($\geq 1$) transactions $T_1, \ldots, T_{tn}$. Each transaction $T_k$ is a sequence of read and write operations on objects in the object set $O$. Here, $mtan$ ($\geq 1$) is the maximum number of operations in each transaction $T_k$. Each transaction $T_k$ is composed of $tan_k$ ($1 \leq tan_k \leq mtan$) operations. The number $tan_k$ for each transaction $T_k$ is randomly selected out of 1, 2, …, $mtan$. Each transaction $T_k$ is granted one role $p_k$ which is randomly selected in the role set $R$. Each operation $op$ on an object $o_i$ is randomly selected in the role $p_k$ of the role set $R$, i.e. $\langle o_i, op\rangle \in p_k$ so that duplicate operations may be included. Here, an operation type $op$ is randomly selected in $rd$ and $wr$. Let $\rho$ be a ratio of the number of read operations to the total number of operations. In addition, for each write $op$, full and partial write types are randomly selected, i.e. the ratios of full write operations and partial write operations are (1 - $\rho$) / 2.

A sequence $T$ of the transactions $T_1, \ldots, T_{tn}$ are serially performed on the object set $O$ given the role set $R$ in the WA, RWA, FRWA, and NBS protocols. In the NBS protocol, if a transaction reads data in an object, there might occur illegal information flow as discussed. Let $nir$ be the number of illegal read operations and $npw$ be the number of impossible write operations. $npw$ shows how much illegal information flow occurs from suspicious objects. Let $t\_rabort$ and $t\_oabort$ be numbers of transactions aborted in the RBS type and OBS type of protocols, respectively, where $t$ shows a type of synchronization, $t \in$ {WA, RWA, FRWA}.

In the evaluation, first, a collection $R$ of roles $r_1, \ldots, r_{rn}$ are generated by randomly selecting access rights on the objects $o_1, \ldots, o_n$, $R = \{r_1, \ldots, r_{rn}\}$ for given numbers $n$ of objects and *rn* of roles. Then, a sequence $T$ of the *tn* transactions $T_1, \ldots, T_{tn}$ are generated by randomly selecting operations and objects on the object set $O$ with the role set $R$ for a given number *mtan* of transactions. Here, $10 \leq mtan \leq 200$. The sequence $T$ of the transactions are serially performed on the object set $O$ in the protocols.

We randomly create a role set $R$ and a transaction sequence $T$ on the object set $O$ three hundreds times for each *mran*. For a given role set $R$ and each of the WA, RWA, FRWA, and NBS protocols, the transaction sequence $T$ is performed five hundreds times. Then, we calculate the average values of *t-rabort*, *t-oabort*, *nif*, and *npw* for the RBS, OBS, and NBS protocols, respectively, where $t$ stands for WA, RWA or FRWA protocols.

### *5.2 Evaluation results*

First, the abortion probability *ap* of each transaction $T$ is 1 / 2 in the FRWA protocols. Once issuing an illegal or suspicious read operation to an object, a transaction is aborted with probability 1 / 2. The WA, RWA, FRWA, and NBS protocols are evaluated on a role set $R$ and a transaction sequence $T$.

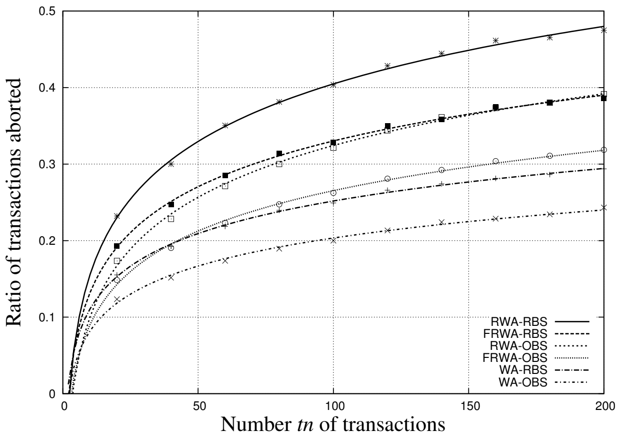


Fig. 6. Number of transactions aborted for $\mu = 0.1$ and $\rho = 0.5$.

Figure 6 shows the ratios of the number of transactions aborted to the total number *tn* of transactions where suspicious ratio $\mu = 0.1$ and read ratio $\rho = 0.5$ in the WA, RWA, and FRWA protocols. The number of transactions aborted in the FRWA protocols is fewer than the RWA protocols but more than the WA protocols. For example, about 20% and 32% of the transactions are aborted in the WA-OBS and RWA-OBS protocols, while about 26% of the transactions are aborted in the FRWA-OBS protocol for one hundred transactions ($tn = 100$).

In the WA protocols, meaningless read operations are performed since transactions are not aborted even if the transactions issue illegal read operations. Figure 7 shows the ratios of the number of meaningless read operations in the WA and FRWA protocols for suspicious ratio $\mu = 0.1$ and read ratio $\rho = 0.5$. In the WA and FRWA protocols, the number of meaningless read operations increases as the number *tn* of transactions increases. The number of meaningless read operations in the FRWA protocols is fewer than the WA protocols. For example, about

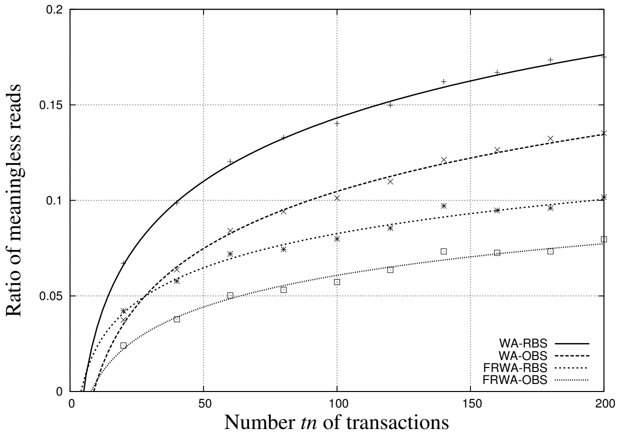


Fig. 7. Number of meaningless read operations for $\mu = 0.1$ and $\rho = 0.5$.

11% and 14% of read operations are meaningless in the WA-OBS and WA-RBS protocols, while about 8% in the FRWA-RBS protocol for one hundred transactions $tn = 100$. In the FRWA-OBS protocol, the ratio of meaningless read operations is the smallest in the protocols. The number of meaningless read operations in the FRWA-OBS protocol is about 43% and 55% of the WA-RBS and WA-OBS protocols, respectively. This means, transactions are more efficiently performed in the FRWA protocols than the WA protocols since a fewer number of read operations, i.e. meaningless read operations are performed in the FRWA protocols than the WA protocols.

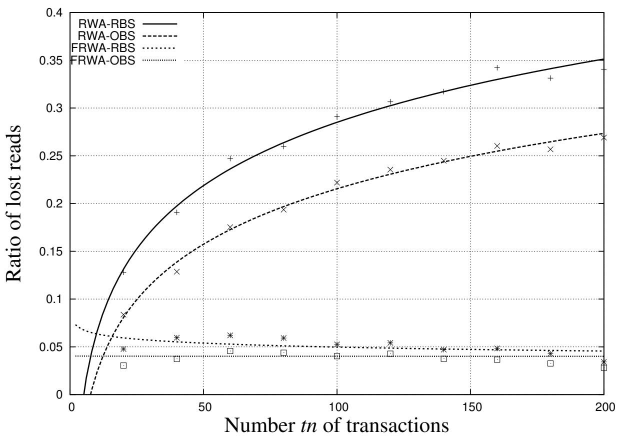


Fig. 8. Number of lost read operations for $\mu = 0.1$ and $\rho = 0.5$.

In the FRWA protocols, some read operations which can be performed without illegal information flow are not performed, i.e. lost read operations since transactions are aborted once issuing illegal read operations. Figure 8 shows the ratios of the number of lost read operations in the RWA and FRWA protocols for suspicious ratio $\mu = 0.1$ and read ratio $\rho$

= 0.5. In the RWA protocols, the number of lost read operations increases as the number of transactions increases. In the FRWA-RBS protocol, the number of lost read operations decreases as the number $tn$ of transactions increases. In the FRWA-OBS protocol, the number of lost read operations does not change even if the number $tn$ of transactions increases. There are fewer number of lost read operations in the FRWA protocols than the RWA protocols. For example, about 22% and 29% of read operations are lost in the RWA-OBS and RWA-RBS protocols, while about 5% are lost in the FRWA-RBS protocol for one hundred transactions ($tn = 100$). In the FRWA-OBS protocol, the number of lost read operations is the smallest. In the RWA protocols, there are more number of lost read operations which can be performed but are not performed due to transaction abortion than the FRWA protocols.

In the Table 2, the abortion ratio of transactions and numbers of meaningless read operations and lost read operations are summarized. The more number of transactions are aborted in the FRWA protocols than the WA protocols. However, the execution time of each transaction is shorter in the FRWA protocols than the WA protocols since there are a fewer number of meaningless read operations. The number of lost read operations which can be performed but is not performed in the FRWA protocols are fewer than the RWA protocols. This means, more number of read operations can be performed in the FRWA protocols than the RWA protocols.

Table 2. Meaningless and lost read operations.

| protocols | abortion ratio | meaningless read | lost read |
|---|---|---|---|
| WA | △ | ○ | — |
| RWA | ◇ | — | ○ |
| FRWA | ○ | △ | △ |

— : not exist, △ is smaller than ○ (△ < ○), ○ < ◇.

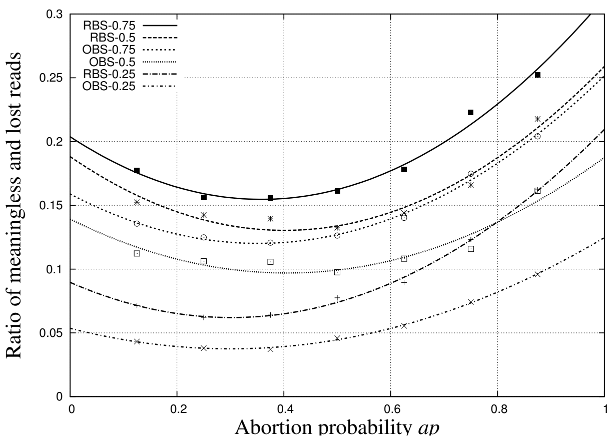


Fig. 9. Number of meaningless and lost read operations for $tn = 100$ and $\mu = 0.1$.

We measure the total number of meaningless and lost read operations in the FRWA protocols. The smaller number of meaningless and lost read operations, the better in terms of the performance. We discuss how the number of meaningless and lost read operations changes as the abortion probability $ap$ changes. Figure 9 shows the ratios of the number of meaningless and lost read operations to the total number of read operations in the FRWA-RBS and FRWA-OBS protocols for read ratio $\rho = 0.25, 0.5, 0.75$ with the number of transactions $tn = 100$ and suspicious ratio $\mu = 0.1$. The number of meaningless and lost read operations is the smallest where the abortion probability is about 0.3 to 0.5 for any read ratios $\rho$. For example, the ratio of meaningless and lost read operations is minimum with abortion probability $ap = 0.41$ in the FRWA-OBS protocol for read ratio $\rho = 0.5$. In the FRWA protocols, we can take the abortion probability $ap$ is 0.3 to 0.5 for any read ratio $\rho$.

## 6 Concluding Remarks

In this paper, we discussed how to prevent illegal information flow on the role-based access control (RBAC) model. In our previous studies [12, 14, 15], the WA-RBS, WA-OBS, RWA-RBS, and RWA-OBS protocols are proposed where illegal information flow is prevented by aborting transactions. In the WA protocols, a transaction is aborted once issuing an illegal or impossible write operation. A transaction is not aborted even if the transaction issues an illegal read operation to an object. In the RWA protocols, a transaction is aborted only if the transaction issues an illegal read operation or impossible write operation to an object. In this paper, we newly proposed the FRWA-RBS and FRWA-OBS protocols where a transaction is aborted with some probability if the transaction issues an illegal read operation to an object. In the evaluation, the number of transactions aborted in each type of the FRWA protocols is fewer than the RWA protocols but larger than each type of the WA protocols. A fewer number of meaningless read operations are issued in the FRWA protocols than the WA protocols. This means, the execution time of each transaction is shorter in the FRWA protocols than the WA protocols. In addition, a fewer number of lost read operations are issued in the FRWA protocols than the RWA protocols. That is, a more number of read operations can be performed than the RWA protocols in the FRWA protocols. We also showed the abortion probability $ap$ can be taken $0.3 - 0.5$.

## Acknowledgements

This work was supported by JSPS KAKENHI grant number 15H0295.

## References

1. J. Bacon, D. Eyers, T. F. J. -M. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch (2014), *Information Flow Control for Secure Cloud Computing*, IEEE Transactions on Network and Service Management, Vol.11, No.1, pp.1-14.
2. D. E. R. Denning (1982), *Cryptography and Data Security*, Addison Wesley, 400 pages.
3. T. Enokido and M. Takizawa (2009), *A Legal Information Flow (LIF) Scheduler Based on Role-based Access Control Model*, International Journal of Computer Standard and Interfaces, Vol.31, No.5, pp.906-912.
4. T. Enokido and M. Takizawa (2010), *A Purpose-based Synchronization Protocol for Secure Information Flow Control,* International Journal of Computer Systems Science and Engineering, Vol.25, No.2, pp.25-32.
5. T. Enokido and M. Takizawa (2011), *Purpose-based Information Flow Control for Cyber Engineering*, IEEE Transactions on Industrial Electronics, Vol.58, No.6, pp.2216-2225.

6. E. B. Fernadez, R. C. Summers, and C. Wood (1980), *Database Security and Integrity*, Addison Wesley, 319 pages.
7. D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli (2007), *Role-based Access Control (2nd ed.)*, Artech, 381 pages.
8. K.-S. Fisher-Hellmann (2012), *Information Flow Based Security Control Beyond RBAC*, Springer Vieweg, 159 pages.
9. J. Gray and A. Reuter (1993), *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1070 pages.
10. C. Hammer and G. Snelting (2009), *Flow-sensitive, Context-sensitive, and Object-sensitive Information Flow Control Based on Program Dependence Graphs*, International Journal of Information Security, Vol.8, No.6, pp.399-422.
11. S. Nakamura, D. Duolikun, and M. Takizawa (2015), *Read-abortion (RA) Based Synchronization Protocols to Prevent Illegal Information Flow*, Journal of Computer and System Science, Vol.81, No.8, pp1441-1451.
12. S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa (2015), *A write abortion-based protocol in role-based access control systems*, International Journal of Adaptive and Innovative Systems, Vol.2, No.2, pp.142-160.
13. S. Nakamura, D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa (2014), *Role-based Information Flow Control Models*, Proc. of IEEE the 28th International Conference on Advanced Information Networking and Applications (AINA-2014), pp.1140-1147.
14. S. Nakamura, D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa (2014), *Synchronization Protocols to Prevent Illegal Information Flow in Role-based Access Control Systems*, Proc. of International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2014), pp.279-286.
15. S. Nakamura, D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa (2014), *Read-Write Abortion (RWA) Based Synchronization Protocols to Prevent Illegal Information Flow*, Proc. of International Conference on Network-Based Information Systems (NBiS-2014), pp.120-127.
16. S. Osborn, R. S. Sandhu, and Q. Munawer (2000), *Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies*, ACM Transactions on Information and System Security, Vol.3, No.2, pp.85-106.
17. R. S. Sandhu (1993), *Lattice-based Access Control Models*, IEEE Computers, Vol.26, No.11, pp.9-19.
18. R. S. Sandhu (1996), *Role-based Access Control Models*, IEEE Computers, Vol.29, No.2, pp.28-47.
19. M. Yasuda, T. Tachikawa, and M. Takizawa (1998), *A Purpose-Oriented Access Control Model for Information Flow Management*, Proc. of the 14th IFIP International Information Security Conference (IFIP/SEC'98), pp.230–239.
20. N. Zeldovich, S. Boyd-Wickizer, and D. Mazieres (2008), *Securing Distributed Systems with Information Flow Control*, Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp.293-308.