# The Bell-LaPadula Computer Security Model Represented as a Special Case of the Harrison-Ruzzo-Ullman Model

Paul A. Pittelli

## ABSTRACT

Currently most computer security models are classified among the three types; access control, information flow, and non-interference. Within the realm of access control lies the classical Bell-LaPadula model. A BLP model consists of a set of subjects and objects, three security level functions, and a discretionary access matrix together with a set of rules used to manipulate the current state of the model. Security in this model is dependent upon the satisfaction of the three properties: simple security, discretionary access, and the * property. An HRU model consists of an access matrix and a finite set of commands which act as matrix transformations. Here security is determined by looking for the existence of an access right in a specific cell of the matrix. We define a specific HRU model (called the Bobo model) and establish a correspondence between the Bobo commands and BLP rules, also between the Bobo and BLP states. Furthermore we observe that this correspondence is security preserving in the fact that a BLP access triple is secure if and only if that access is contained in a specific cell of the Bobo access matrix.

## Introduction

The purpose of this note is to show that the Bell-LaPadula model for access control is simply a special case of the not so well known Harrison-Ruzzo-Ullman model. The HRU model consists of an access matrix together with a finite set of commands that are used to manipulate the matrix. In order to develop a model equivalent to BLP's, we need to exhibit commands that are "identical" to the BLP rules.

Before we begin defining the commands, we must first exhibit a correspondence between the "subjects" and "objects" in the BLP model and the "subjects" and "objects" in the HRU environment. The reason for the above quotes is that subjects and objects are disjoint sets in BLP, whereas the set of subjects is contained in the set of objects under HRU. This distinction, though seemingly small, is well worth remembering. However, a key factor of the BLP model is the use of the functions $f_S$,$f_C$,and $f_O$. These functions associate a security value to a subject or object, which allows one to compare subjects to objects.

The preceeding paragraph indicates some of the differences we need to consider when relating BLP to HRU. The major concept in this new model is the notion of a subject(object) represented by a set of entities. Defining a subject as a set of sub-subjects allows us to implement the multilevel capabilities of a BLP subject in an access matrix. Likewise an object viewed as a set permits the use of an upgrade command.

Specifically, suppose we have a BLP model which consists of the following entities;

$\Sigma = \{s_i: i=1,2,\ldots,k\}$ -- set of subjects

$\hat{O} = \{o_j: j=1,2,\ldots,n\}$ -- set of objects; recall $\Sigma \cap \hat{O} = \emptyset$

$L = \{l_t: t=0,1,\ldots,T\}$ --set of security values forming a lattice under the partial ordering $\gg$ referred to as the dominance relation. Without loss of generality let $l_0$ and $l_T$ denote the least upper bound of L and the greastest lower bound of L respectively.

$f_S: \Sigma \Rightarrow L$ -- function yielding the maximum security level for a subject.

$f_C: \Sigma \Rightarrow L$ -- function yielding the current security level for a subject.

$f_O: \hat{O} \Rightarrow L$ -- function yielding the security value of an object.

R = {append,write,read,execute} -- set of access rights.

$M = k \times n$ matrix with $m_{ij} \subseteq R$ representing the set of discretionary access rights that subject $s_i$ has to object $o_j$.

We now begin showing how to incorporate the BLP model into an HRU environment. First we define the following components of an HRU model:

$S = \{s_i l_t: s_i \varepsilon \Sigma \text{ and } t \varepsilon ST_i\} \cup \{s_0 l_T\}$.Corresponding to each BLP subject $s_i$ is a subset $S_i$ of S, where $S_i = \{s_i l_t: t \varepsilon ST_i\}$, which represents $s_i$ together with all of $s_i$'s allowable security values. That is $\{l_t: t \varepsilon ST_i\} = \{l_t \varepsilon L: f_S(s_i) \gg l_t\}$. Formally the elements of $S_i$ come from the cross product space $\Sigma \times \hat{O}$, but for ease of notation we will write the elements as $s_i l_t$. Thus $s_i l_t$ will denote an HRU subject. Furthermore we reserve the subject $s_0 l_T$ to be a system subject. Thus $S_0 = \{s_0 l_T\}$ and $ST_0 = \{T\}$. The purpose of $s_0 l_T$ is to let the system know at what level an object is currently classified as will be formalized later.

$O = S \cup \hat{O}$ where $\hat{O} = \{o_j l_u: o_j \varepsilon \hat{O} \text{ and } u \varepsilon OT_j\}$.We relate to each object $o_j$ a set of values $\{l_u: u \varepsilon OT_j\}$ which represents all the security values that $o_j$ could assume. That is $\{l_u: u \varepsilon OT_j\} = \{l_u \varepsilon L: l_u \gg f_O(o_j)\}$.

A = {active,own,r,a,w,e}set of generic access rights.

$P = (P[s,o])$, $p \times (p+q)$ matrix where $P[s,o] \subseteq A$ for $s \varepsilon S$ and $o \varepsilon O$. Here

$$p = \sum_{i=0}^{k} \left| ST_i \right| \; and\, q = \sum_{j=1}^{n} \left| OT_j \right| .$$

P is simply referred to as the access matrix.

## Primitive Operations:

In the BLP model the rules allow for the creation and deletion of objects as well as the insertion and removal of an access right from a cell in the access matrix M. In the HRU model there are counterparts of these actions which are termed primitive operations. Because of our particular example there is an additional operation: delete a proper subset of the set $O_j = \{o_j l_u : u \; \varepsilon \; OT_j\}$. This last primitive operation will provide the means to implement an upgrade command.

Given system state (S,O,P) we define a primitive operation op as a function $op:(S,O,P) \Rightarrow (S^*,O^*,P^*)$ where:

(1) op = create object $o_j l_u$,

where $O_j \not\subset O$. We have for all $l \gg l_u$;

$S^* = S, O^* = O \cup \{o_j l\}$,

$P^*[s,o] = P[s,o]$ for all $(s,o) \; \varepsilon \; S \times O$

$P^*[s,o_j l] = \varnothing$ for all $s \; \varepsilon \; S$.

(2) op = delete object $O_j$,

where $O_j \subseteq O \setminus S$. We have for all $l \; \varepsilon \; L$;

$S^* = S, O^* = O \setminus \{o_j l\}$,

$P^*[s,o] = P[s,o]$ for all $(s,o) \; \varepsilon \; S \times O^*$.

(3) op = delete objects $o_j l_u$,

where $o_j l_u \; \varepsilon \; O \setminus S$. We have for all $l \not\gg l_u$,

$S^* = S, O^* = O \setminus \{o_j l\}$,

$P^*[s,o] = P[s,o]$ for all $(s,o) \; \varepsilon \; S \times O^*$.

(4) op = enter x into $P[S_i,o_j l_u]$,

where $x \; \varepsilon \; A$, $S_i \subseteq S$, and $o_j l_u \; \varepsilon \; O \setminus S$. We have for all $l, l_t$ with

$$l_u \gg l \text{ and } \begin{cases} l_t \gg l & \text{if} \quad x = r \\ l_t = l & \text{if} \quad x = w \\ l \gg l_t & \text{if} \quad x = a \\ l_t = l_T & \text{if} \quad x = active \\ \varnothing & \text{if} \quad x = e \text{ or } own \end{cases}$$

$S^* = S, O^* = O$,

$P^*[s,o] = P[s,o]$ for all $(s,o) \neq (s_i l_t, o_j l)$

$P^*[s_i l_t, o_j l] = P[s_i l_t, o_j l] \cup \{x\}$.

(5) op = delete x from $P[S_i,o_j l_u]$,

where $x \; \varepsilon \; A$. We have for all $l, l_t$ with $l_u \gg l$,

$S^* = S, O^* = O$,

$P^*[s,o] = P[s,o]$ for all $(s,o) \neq (s_i l_t, o_j l)$

$P^*[s_i l_t, o_j l] = P[s_i l_t, o_j l] \setminus \{x\}$.

As an aid to understanding the effects of the primitive operations on the matrix P, it is helpful to consider that there corresponds to each subject,object pair $(s_i,o_j)$ a submatrix $P_{ij}$ whose rows are indexed by $ST_i$ and whose columns are indexed by $OT_j$. The consequences of applying the primitive operations can be summarized as follows:

(1) op = create object $o_j l_u$

This operation creates a set of matrices $\{P_{ij}: o \le i \le n\}$, where $P_{ij}$ has rows corresponding to elements in $ST_i$ and columns corresponding to the members of $OT_j$. The cells of each submatrix are all empty.

(2) op = delete object $O_j$

This operation removes from the matrix P all those submatrices $P_{ij}$ for $0 \le i \le k$. Recall that k is the cardinality of S, the set of BLP subjects.

(3) op = delete objects $o_j l_u$

This operation removes a subset of columns from each matrix $P_{ij}$, $0 \le i \le k$, specifically all those columns corresponding to $o_j l$ where $l \not\gg l_u$.

(4) op = enter x into $P[S_i,o_j l_u]$

This operation inserts x into a subset of positions of the matrix $P_{ij}$ defined by the various values of x.

(5) op = delete x from $P[S_i,o_j l_u]$

This operation deletes x from all entries in those columns of $P_{ij}$ corresponding to l where $l_u \gg l$.

## Commands:

An HRU command is simply a conditional IF (expression 1) THEN (expression 2) where expression 1 is a boolean function and expression 2 is a sequence of primitive operations. To implement the BLP model in an HRU environment we will use the following commands. For ease of notation let $S_r$ = requesting subject and x a member of the set {r,a,w,e}.

(1) Command GIVE($S_r,S_i,x,o_j l_u$)

IF own $\varepsilon$ $P[s_r l_t, o_j l_u]$ for any $l_t$, and active $\varepsilon$ $P[s_0 l_T, o_j l_u]$

THEN enter x into $P[S_i, o_j l_T]$.

(2) Command RESCIND($S_r,S_i,x,o_j l_u$)

IF own $\varepsilon$ $P[s_r l_t, o_j l_u]$ for any $l_t$

THEN delete x from $P[S_i, o_j l_u]$.

(3) Command GENERATE($S_r,o_j l_u$)

IF TRUE

THEN create object $o_j l_u$,
enter active into $P[S_0, o_j l_u]$,
enter own into $P[S_r, o_j l_T]$.

(4) Command DESTROY($S_r,o_j l_u$)

IF own $\varepsilon$ $P[s_r l_t, o_j l_u]$ for some $l_t$

THEN delete object $O_j$.

(5) Command UPGRADE($S_r, o_j, l_{u_0}, l_{u_1}$)

IF own $\varepsilon$ $P[s_r l_t, o_j l_{u_0}]$ for some $l_t$, and active $\varepsilon$ $P[s_0 l_T, o_j l_{u_0}]$

THEN delete objects $o_j l_{u_1}$,
enter active into $P[S_0, o_j l_{u_1}]$.

Note: For the rest of this paper the sets S, O, access matrix P, and the five commands defined above will comprise that which will be called the Bobo model.

## Equivalence to BLP:

The method that we will use to exhibit an equivalence between the BLP and Bobo models is a two-fold process First we will prove a theorem that will show every state of a BLP model is achievable by the Bobo model. Secondly a correspondence between the state transitions of the two models will be drawn by simply listing each BLP state transition together with its counterpart Bobo state transition.

Besides showing that every BLP state is achievable by the Bobo model, the theorem below illustrates how to identify the BLP set of secure access triples in the HRU environment. One of the hypotheses of the theorem is an assumption on the initial access matrix $P^0$. We assume that for $0 \le i \le k$; $1 \le j \le n$; $t \varepsilon ST_i$; $u \varepsilon OT_j$;

$$P^0[s_i l_t, o_j l_u] = \begin{cases} \{active\} & if\ s_i = s_0\ and\ l_u = f_O(o_j) \\ \{own\} & if\ s_i\ created\ o_j \\ \{\} & otherwise \end{cases}$$

In order to see that this is not an unreasonable assumption, consider what happens when we generate an object $o_j$. Suppose in the BLP model subject $s_i$ created object $o_j$ at level $f_O(o_j) = l_u$. In the Bobo model this is accomplished by issuing the command GENERATE($S_i$,$o_j l_u$). Upon execution of the command we see that the submatrices $P_{aj}$ are all empty except when $a = i$ in which the matrix $P_{ij}$ contains {own} in every cell. Also there is only one entry in $P_{0j}$ which is nonempty and that is $P^0[s_0 l_T, o_j f_o(o_j)] = \{active\}$. Thus we see that if the system knows who created each object then we can generate the initial matrix $P^0$ by a sequence of GENERATE commands. Hence our assumption on the matrix $P^0$ can be made without loss of generality.

**Theorem**: Let M,f be a state of a BLP model with subjects $\{s_1,s_2,...,s_m\}$ and objects $\{o_1,o_2,...,o_n\}$. Let $\beta$ be the set of all possible secure triples (s,o,x) completely determined by M and f. Define a Bobo access matrix $P^0$ to be, for $0 \le i \le k; 1 \le j \le n$; $t \varepsilon ST_i$; $u \varepsilon OT_j$;

$$P^0[s_i l_t, o_j l_u] = \begin{cases} \{active\} & if\ s_i = s_0\ and\ l_u = f_O(o_j) \\ \{own\} & if\ s_i\ created\ o_j \\ \{\} & otherwise \end{cases}$$

Then there exists a sequence of commands $c_1,c_2,...,c_k$ such that $(S^0,O^0,P^0) \Rightarrow (S^1,O^1,P^1) \Rightarrow ... \Rightarrow (S^k,O^k,P^k)$ and $(s,o,x)\ \varepsilon\ \beta \Leftrightarrow x\ \varepsilon\ P^k[sf_C(s),of_O(o)]$.

<u>Pf</u>: For every object $o_j$ we perform the following:
Suppose $s_d$ is the creator of $o_j$,
then for all $x\ \varepsilon\ m_{ij}$ issue the command GIVE($S_d$,$S_i$,x,$o_j f_O(o_j)$), for $i = 1,2,...,m$.
Since the set of subjects, objects, and access rights are finite sets then we have generated a finite sequence of commands say $c_1,c_2,...c_k$ which transforms $(S^0,O^0,P^0) \Rightarrow (S^k,O^k,P^k)$.
Claim: $(s,o,x)\ \varepsilon\ \beta \Leftrightarrow x\ \varepsilon\ P^k[sf_C(s),of_O(o)]$.
<u>Pf</u>: Suppose$(s_i,o_j,x)\ \varepsilon\ \beta \Leftrightarrow$

$$x\,\varepsilon\, m_{ij}\ and \begin{cases} f_S(s_i) \geqslant f_O(o_j)\ and\ f_C(s_i) \geqslant f_O(o_j)\ if\ x = r \\ f_S(s_i) \geqslant f_O(o_j)\ and\ f_C(s_i) = f_O(o_j)\ if\ x = w \\ f_O(o_j) \geqslant f_C(s_i)\ if\ x = a \\ if\ x = e \end{cases} \Leftrightarrow$$

GIVE($S_d$,$S_i$,x,$o_j f_O(o_j)$) is executed, where $s_d$ is the creator of $o_j \Leftrightarrow$
$x\ \varepsilon\ P^k[s_i f_C(s_i),o_j f_O(o_j)]$. ●

Now we need to show that all the possible state transitions in the BLP model are attainable in this new setting. Referring to [1] we find that there are 11 state transitions(rules). For each rule we will establish an equivalent command and/or a reason why the rule is satisfied.

Rules 1-5: get,release read/append/write/execute

It is unnecessary to implement a get or release command in our Bobo model. In BLP, a subject has to request get access to an object so that the * property is never violated. However in the Bobo model the enter access primitive operation assures that the * property will not be violated. Thus a subject in the Bobo model obtains access to an object whenever the owner of the object has given him the desired access by executing a GIVE command.

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

This rule corresponds to the command GIVE. The rule checks to see if the requesting (giving) subject has the authority to "give" another subject access to a specific object. This is accomplished by the command via the condition that the requesting subject have"ownership" of the object in question. Furthermore upon a true condition, the given access right is granted so that the BLP * property is not violated. (e.g. If $s_i$ is granted w access to $o_j$, then w is only added to the set in positions $(s_i l_t, o_j l_t)$ for all $t\ \varepsilon\ OT_j$.)

Rule 7: rescind - read/append/write/execute

The command RESCIND performs the inverse of GIVE as does the rule rescind in the BLP sense. Again the condition tests the authority of the requesting subject via "ownership" and upon valid authority removes the specified access from the subject's access columns.

Rule 8,9 create,delete object

Commands GENERATE and DESTROY correspond to the rules create and delete respectively. GENERATE creates an object and defines the initial "owner" of the object to be its creator. DESTROY checks for"ownership" for authority to delete the object from the system.

Rule 10: change-subject-current-security-level

There is no need for a command which changes a subject's current security level. The reason is that the Bobo model defines a subject $S_i$ to be the set $\{s_i l_t: t\ \varepsilon\ ST_i\}$. This allows a subject to work on an object $o_j l_u$ in mode x if and only if there is some $t\ \varepsilon\ ST_i$ such that $x\ \varepsilon\ P[s_i l_t, o_j l_u]$. Thus the command automatically changes a subjects current security level to accomodate the desired access to an object.

Rule 11: change-object-security-level

The command UPGRADE performs the function of changing the security level of an object. In BLP the reference monitor needs to verify that the new level is a valid one and that the requesting subject has the authority to change the function $f_O$. This is accomplished by the "own" access right and the delete objects primitive operation. Furthermore suppose that object $o_j$ gets upgraded from level $l_u$ to $l_v$. Notice that because of the primitive operation enter, if subject $s_i$ had access x to $o_j l_u$ then $s_i$ will still have access x to $o_j l_v$ provided that the * property is not violated. This implies that the UPGRADE command automatically cancels all current accesses that violate the * property at an objects new security level.

## <u>Remarks:</u>

Even though the Bobo model can simulate the BLP model, there are several characteristics that have been created to accomplish this. The first and most important is the new idea of an access right called "own". The BLP model contains a tree structure for the objects called a hierarchy. However the hierarchy of objects is not related to the security of the model but exists merely because of the application of BLP to the Multics system. Thus we see that the only concept of ownership of an object in BLP lies in the Give function for rule 6. In order to implement this Give function it is necessary to use an "own" access right. The Bobo model is conservative in the fact that the only subject who can have "own" access to an object is that object's creator. However if there is need for group ownership of an object the conditions of the GIVE command can be changed to accommodate this feature.

The other new access right in the Bobo model is "active". The purpose of "active" is simply to let the reference monitor know the current level of an object. This feature then allows for the upgrading of an objects security level.

The use of sets for representing subjects and objects creates more responsibilities for the reference monitor in the Bobo model. In particular, since a subject can work at any level he dominates and an object can assume various security values, then it should be the job of the reference monitor to inform the subject at what level he is working and the value of the object that he is accessing.

The last remark that we want to make concerns the form of the access matrix P. The easiest observation to make is that the submatrix whose rows and columns are indexed by the subjects is completely empty. This reflects the fact that subjects are not allowed to access other subjects in the BLP model. Also the method of giving subjects access rights creates alot of redundant information. That is, if the reference monitor wants to check to see if subject $s_i$ has access x to object $o_j l_u$ then the only cell necessary to examine is $P[s_i l_u, o_j l_u]$.

## Safety:

This section discusses the concept of safety as introduced by Harrison, Ruzzo and Ullman. Intuitively a "safe" security model (i.e. protection system in HRU terminology) is one which will not allow unauthorized access to objects. The following two definitions formally state the idea of safety.

**Def:** Given a protection system, we say a command $\propto$ *leaks generic right r* from configuration Q = (S, O, P) if $\propto$, when run on Q, can execute a primitive operation which enters *r* into a cell of the access matrix which did not previously contain *r*.

**Def:** Given a particular protection system and generic right *r*, we say that the initial configuration *$Q_o$ is unsafe for r* (or leaks r) if there is a configuration Q and a command $\propto$ such that

(1) $Q_o \Rightarrow Q$ by a sequence of primitive operations,

(2) $\propto$ leaks *r* from Q.

The first observation one can make is that any system which utilizes the primitive operation enter will most likely be deemed unsafe. Harrison et al. make the convention that to check for unauthorized leakage, we need to eliminate from testing those subjects who are actually authorized to give (leak) rights. They use the term "reliable" subjects to mean the set of subjects who are authorized to grant the generic right *r* of an object to another subject.

The general question of whether or not an arbitrary protection system is safe was shown to be undecidable by HRU. However if a specific model consists of commands which involve only one primitive operation (mono-operational in HRU terminology) then the question of safety is decidable. Our Bobo model happens to live in the middle ground of the previous sentences. That is, the Bobo model is not a mono-operation system but we will show that the model is "safe".

Its not too hard to see that every command except the GENERATE command contains a check for ownership in the condition. Since a subject who owns an object is deemed reliable then the only command in question is the GENERATE command. However, anyone who creates an object is by nature reliable with respect to that object. So we can view the entering of the own access right by the creator of the object giving himself access. Therefore, all the commands in the Bobo model preserve safety which implies the Bobo model is itself "safe".

## Conclusions:

Besides an attempt at unifying some of the existing access control models, the Bobo model reveals an interesting point. This is we see that the HRU model is a very general access control model. Moreover one can appreciate the elegance of the model by the fact that the complex BLP model can be defined by an access matrix together with a set of five commands. Being able to incorporate the three functions $f_s$, $f_c$, $f_o$, the BLP discretionary access matrix M, and the set of all current accesses b into the matrix P, allows one to see the interplay between the different security levels and the * property. Also note that we have only dealt with the rules defined by volume 4 of the Bell-LaPadula model. A current topic of discussion is whether or not the rules constitute a part of a BLP model. This is no concern of this paper so we do not attempt to imply either case. However what we show is how the rules of any BLP model correspond to commands in an HRU model. Thus if any more rules are added to the existing BLP model then a new command can be added to the Bobo model accordingly in order to preserve the equivalence. Even though implementation of the BLP model might be simplified by using the Bobo model, the intent of this paper and follow-ons is to try to unite all the existing access control models in one framework; maybe that of the Harrison, Ruzzo, Ullman Model.

**References:**

(1) Bell, D.E. and LaPadula, L. J. Secure Computer Systems, Vol. IV: Unified Exposition and Multics Interpretation. MITRE Corp. Tech. Rep. MTR 2997, 1975.

(2) Harrison, M. A., Ruzzo, W.L. and Ullman, J. D. Protection in Operating Systems. Communications of the ACM, Vol. 19, No. 8, August 1975