Towards A Times-based Usage Control Model

Baoxian Zhao[1], Ravi Sandhu[2], Xinwen Zhang[3], and Xiaolin Qin[4]

[1] George Mason University, Fairfax VA, USA
bzhao@gmu.edu
[2] Institute for Cyber-Security Research, Univ. of Texas at San Antonio, USA
ravi.sandhu@utsa.edu
[3] Samsung Information Systems America, San Jose, CA, USA
xinwen.z@samsung.com
[4] Nanjing University of Aeronautics and Astronautics, Nanjing, China
qinxcs@nuaa.edu.cn

Abstract. Modern information systems require temporal and privilege-consuming usage of digital objects. To meet these requirements, we present a new access control model–Times-based Usage Control (TUCON). TUCON extends traditional and temporal access control models with times-based usage control by defining the maximum times that a privilege can be exercised. When the usage times of a privilege is consumed to zero or the time interval of the usage is expired, the privilege exercised on the object is automatically revoked by the system. Formal definitions of TUCON actions and rules are presented in this paper, and the implementation of TUCON is discussed.

Key words: Access Control, Usage Control, Times-based Usage Control, TUCON, Authorization

1 Introduction

The rapid development in information technology, especially in electronic commerce applications, requires additional features for access control. In recent information systems, usage of a digital object can be not only time-independent like read and write, but also temporal and times-consuming, such as payment-based online reading metered by reading times or chapters, or a downloadable music file that can only be played 10 times. In these applications, the access to an object may decrease, expire, or be revoked along with the usage times of the object.

Traditional and temporal access control models are not suitable for the above requirements since the authorization decisions in these models are generally made at the requested time but hardly recognize ongoing controls for times constrained access or for immediate revocation. In order to meet these requirements in modern access control, this paper presents a new access control model, called Times-based Usage Control (TUCON).

Compared to traditional models, TUCON features with the usage times of privileges and valid periods for usage of digital objects, which enable the ability to express consumed privileges and define their period constraints. The usage times can be triggered by active tasks in TUCON model. For example, once an Internet user pays $10 for an on-line music system, he can enjoy 10 times on-line listening privilege. When a subject is accessing the object in TUCON, the usage times of this subject is decreased by 1. If the times is consumed to zero or the time interval of usage is expired, authorization for the subject is revoked. With the decreasing of usage times, authorizations can be updated during the whole access process, and transferred among users without the problem of privilege chains faced by current and traditional access control. Compared to authorizations in traditional and temporal access control models, authorizations in TUCON are mutable and flexible.

With usage times constraints in TUCON, system resources and privileges can be prevented from being abused. It's known that, through occupying too many resources, some worms and viruses can attack computer systems, such as Code Red [20] and Dukes [17]. If reasonable usage times are given for system resources, such kinds of attacks can be avoided to a great extent [11].

The paper is organized as follows: In Section 2, some related work are discussed. Section 3 shows a simple motivating example, which traditional and temporal access control models cannot support well. Section 4 presents the temporal and times-based constraints in the proposed TUCON model. Authorizations and authorization rules in TUCON are also discussed in this section. In Section 5, we give the implementation of TUCON in practice. Section 6 concludes this paper and presents our future work.

2 Related Work

The development of access control models has experienced a long history. There are two main approaches in this field. One is about traditional access control models. This approach is the earliest research work for access control and originates from the research of discretionary access control (DAC) [2, 6, 12, 21]. A classic paper by Lampson [2] introduced this basic ideas. Because DAC has an inherent weakness that information can be copied from one object to another, it is difficult for DAC to enforce a safety policy and protect against some security attacks. In order to overcome the shortcoming of DAC, mandatory access control (MAC) was invented to enforce lattice-based confidentiality policies [4, 5] in the face of Trojan Horse attacks. MAC does not consider covert channels, but covert channels are expensive to eliminate [22]. Sandhu et al presented Role-based access control (RBAC) [23], which has been considered as a promising alternative to DAC and MAC. There have been much progress in traditional access control, but its core has largely remained unchanged and centered around the access matrix model [2, 3].

The other approach is about the research of temporal access control models, which introduce the temporal attributes into traditional access control with temporal logic. The approach is based on traditional access control. A temporal authorization model was first proposed by Bertino et al in [7], which is based on temporal intervals of validity for authorization and temporal dependencies among authorizations. In [8, 9], Bertino et al extended the range of temporal intervals to temporal periods and suggested an access control model supporting periodicity constrains and temporal reasoning. Following the RBAC model, the Temporal-RBAC (TRBAC) model was presented in [10], which supports temporal dependencies among roles. At the same time, Avigdor et al suggested another authorization model for temporal data called Temporal Data Authorization Model (TDAM) [1]. TDAM extended the basic authorization model by facilitating it with the capability to express authorizations based on the temporal attributes associated with data, such as transaction time and valid time. Recently, TRBAC is extended to the Generalized Temporal RBAC (GTRBAC) in [13], which enables RBAC to express a wider range of temporal constraints. All these temporal access control models primary consider authorization decisions constrained by certain time periods.

Although many researchers in the field of access control have made great contributions to the progress of access control, authorizations in these models are still static authorization decisions based on subjects' permissions on target objects. Once an access to the object is permitted, the subject can access it repeatedly at the valid time intervals.

Sandhu , Park et al have proposed the Usage Control (UCON) [14, 15, 24, 27, 28] model to solve these problems. The UCON model considers this temporal and consumed attributes as the mutable attributes of subjects or objects [15]. The UCON model has unified traditional access control models and temporal access models with its ABC (Authorizations, oBligations and Conditions) [14] core models.

Our approach to solve these problems is different from the work of the UCON model. In TUCON, we focus on periods and usage times of accessing rather than a single access in UCON. In other words, usage times in TUCON is a sequence of accesses. Temporal and consumed authorizations are enforced in TUCON, which make access control simple and easy to be implemented. However, until now, UCON hasn't been put into the practice because the administration of authorizations in UCON is potentially complex and difficult to implement. Therefore, to meet new requirements in the modern access control, TUCON is suitable for applications in the real world.

3 Motivating Example

In this section, a simple example is given to motivate the new features of TUCON. Current and traditional access models have difficulties, or lack flexibility to specify policies in this application.

Consider a simple application with times constrained usage of digital objects, where a registered user can enjoy on-line music for 10 times if only he/she has prepaid $10. Privileges for using objects can be transferred between two registered users and their privileges will be revoked in the following two situations:

1. revocation by usage times: the usage times is zero during ongoing usage of digital objects.
2. revocation by time interval: the time interval of authorizations is expired.

Based on this policy, three different attributes are required to meet these authorizations:

1. The time interval. It includes the starting time and the end time. An access is permitted when the usage times for digital objects is more than zero at the starting time. Otherwise, at the end time, the usage privilege for using objects is revoked.
2. The valid period. Only during the valid period of usage, an access to an object can be permitted, otherwise, denied.
3. Usage times. It is the maximum value which restricts a subject accessing the object. When the usage times of an object is zero, the subject is prohibited to access this object and the privilege is automatically revoked by the system.

Through the above analysis, we give the state transition of times-based usage control actions in Figure 1.

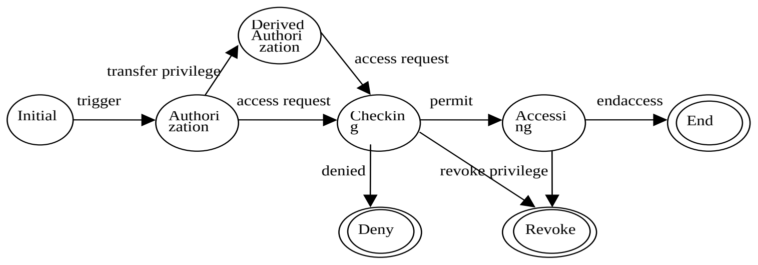

Fig. 1. the state transition of times-based usage control actions.

The states and actions in Figure 1 are explained below.

1. Initial: the initial state of the system.
2. Triggers: triggering authorization of systems, which are requirements that must be satisfied for granting access. In this example, triggers are the actions that a user must register himself in our system and pre-pay \$10 before enjoying this on-line service. Triggers are abstracted into logic expressions in TUCON.
3. Authorization: granting privileges of service to users if users meet authorization requirements of the system.
4. Transfer privilege: one can transfer his privilege of an object to another by decreasing his usage times.
5. Derived authorization: authorizations are derived from transferring privileges or exercising privileges.

6. Access request: the user requests to access digital objects. In section 5, the formalization of an access request is given.
7. Checking: checking the usage times and the period of the authorization .
8. Permitted and denied: If the usage times is more than zero and the time interval is not expired during the valid period , an access to digital objects is permitted, otherwise, denied.
9. Accessing: During this state, subjects are accessing digital objects. During the state of accessing, the usage times of subjects need to be updated by decreasing 1.
10. Revoke privilege and endaccess: If the usage times is not zero and the usage period is till valid after accessing, no updating is done by the system. This case is the action of endaccess. Otherwise, the system will revoke some subjects' privileges and update some authorizations. This process is the action of revoke privilege. The details of revoke privilege are introduced in Section 5.
11. Deny, Revoke and End: three final states. Deny is the sate of refusing to access without revoking privileges. Revoke is the finial state after the action of revoke privileges, while End is the one after the action of endaccess.

From the analysis of states and actions in TUCON, it is obvious that an access is not a simple action, which consists of a sequence of actions and active tasks from a subject and the system. During the whole access process, authorizations need to be updated. These requirements are far out of the scope that traditional and temporal access models can deal with. In the following, TUCON is introduced to solve these problems.

4 TUCON Model

TUCON consists of two aspects: times-based authorizations and authorization rules. Before these aspects are discussed, some preliminaries are given.

4.1 Preliminaries

In order to keep the generality of TUCON and protect information of different data models, no basic data assumption for TUCON is made here. Therefore it is easy to apply TUCON to other data models.

Assume that U denotes the set of subjects (users), O set of objects, P set of privileges for objects, $\mathbb{N}$ set of natural numbers, and T set of time intervals with a total order relation $\leq$.

Definition 1 (Periodic Expression [9]) *A periodic expression is defined as* $Q = \sum_{i=1}^{n} O_i\ .C_i \triangleleft .C_d$, *where* $O_1 \in 2^{\mathbb{N}} \cup \{all\}$, $O_i \in 2^{\mathbb{N}}, i = 2, \ldots, n, C_i$ *and* C_d *are calendars for* $i = 2, \ldots, n, C_d \subseteq C_n$, *and* $d \in \mathbb{N}$.

Let D present the set of all valid periods, then $Q \in D$. Table 1 illustrates a set of periodic expressions and their meanings. In order to simplify the following discussion, all authorization tuples are given with the same privilege on the same object, having the same time interval, and the same period. Since the implementation of operations for periodical data [9,16,18] is not the focus of this paper, any further discussion is not given in this paper.

Definition 2 (Times) *Times is a set of natural numbers, formally defined as* $\{pt \in \mathbb{N}\}$.

4.2 Authorizations

TUCON allows us to express times-based authorizations. That is, authorizations for a user to access an object in specific time intervals are specified by a periodic expression, as well as determined by times of privilege usage. Moreover, the usage times of a privilege is a natural number associated with each authorization, and a time interval is also associated with each authorization, imposing lower and upper bounds to the potentially infinite set of instants denoted by the periodic expression. We refer to an authorization together with its usage times as a *times authorization.*

Definition 3 (Times Authorization) *A times authorization is a 6-tuple (pt, s, o, priv, pn, g), where* $pt \in \mathbb{N}, s, g \in S, o \in O, priv \in P, pn \in \{+, -\}$.

Tuple (*pt, s, o, priv, pn, g*) states that user s has been authorized (if pn = '+') or denied (if pn = '-') for pt times privilege *priv* on object o by user g. For example, the tuple *(6, Tom, Sun, read, +, Sam)* denotes that *Sam* authorizes 6 times privilege *read* on the book *Sun* to *Tom.*

For convenience, the symbol σ is used to project some appointed area of a tuple. For example, with the tuple *A=(pt1, s1, o1, priv1, pn1, g1)*, $\sigma_A(s) = s1$ denotes the s area of the times authorization A, and $\sigma_A(s, g) = (s1, g1)$ denotes a 2-tuple consisting of s and g areas.

In TUCON, all authorizations are uniformly authorized by the system. When transferring privileges, the system can still be regarded as user g, who transfers privileges to other users, since usage times of this user g are correspondingly decreased. Consumed times reduces the transferring capability during transferring privileges. So revoking privileges, we only need to delete the privileges in

Periodical expression	Meaning
$weeks + \{2,6\}.Days$	Tuesday and Saturday
$Months + 15.Days$	15th of every month
$Years + 7.Months \lhd 2.Months$	Summer vacation (July and August of every year)
$Weeks + \{1,...,5\}.Days$	Workday
$Weeks + \{1,...,5\}.Days + 9.Hours \lhd 3.Hours$	Each working day between 9.am and 12 a.m

Table 1. Example of periodic expressions

our system, which doesn't have problems caused by transferring privileges in the current and traditional access control models such as cascading.

Under some conditions, privileges on objects without times constrains are needed. This kind of authorizations is referred as *non-times authorizations.*

Definition 4 (Non-Times Authorization) *When pt = -1 in a times authorization tuple, we call this times authorization as non-times authorization.*

Notice that in TUCON, when *pn* = '-' in an authorization tuple, it states that this authorization is revoked, even though the usage times may not be zero.

Definition 5 (Times-Based Authorization) *A times-based authorization is a 3-tuple* (time, period, auth)*, where time represents a time interval* $[t_a, t_b]$ *,* $0 \leq t_a \leq t_b \in T$*, period is a periodical expression, and auth is a 6-tuple authorization.*

A 3-tuple ($[t_a, t_b]$,*d* ,(*pt*, *s*, *o*, *priv*, *pn*, *g*)) states that user *s* has been authorized (if *pn* = '+') or denied (if *pn* = '-') for pt times privilege *priv* on object *o* by user *g* in the time interval $[t_a, t_b]$ of the period *d*. When *pt*= '-1', TUCON can be reduced to the models discussed in [7–9].

For a times-based authorization ([1/12/2001 ,12/24/2005], *Weeks+2.days*, *(6, Tom, file, read, +, Sam)*), it means that, between Jan. 12 , 2001 and Dec. 24 , 2005, *Tom* has 6 times privilege *read* on object *file*, but he can operate this privilege only on Tuesday each week.

4.3 Authorization Rules

In this section, authorization rules, with similar semantic as [27], are introduced to organize authorizations. We start with the following predicate symbols.

1. A ternary predicate symbol, *access*, an authorization token. The first area of *access* is a time interval *time*, the second is a periodical expression *period*, and the third is a 6-tuple authorization *auth*. The predicate *access* represents authorizations explicitly inserted by the administrator.
2. A ternary predicate symbol, *deraccess*, with the same semantic meaning as *access*. The predicate *deraccess* represents authorizations derived by the system using logical rules of inference.
3. A ternary predicate symbol, *force_ access*, with the same semantic meaning as *access*. The predicate *force_ access* represents authorizations that hold for each subject on each object. It enforces the conflict resolution policy.
4. A symbol $L_i (0 \leq i \leq n)$. It can represent all the above predicate symbols and also trigger expressions required by the system.

First, a grant rule is given to express how to grant subjects authorizations.

Definition 6 (Grant Rule) *A grant rule is defined as the form of:*
access (time, period, auth) $\leftarrow L_1 \& \ldots \& L_n$

where L_i is a trigger condition expression. This expression can be developed from specific requirements for the usage of digital objects. These conditions may be triggers of some active tasks. All these depend on the requirements of the system.

Grant rules are specified to permit accesses to subjects. Whether this rule is true or not is decided by whether or not the condition expressions are satisfied. Note that in TUCON, an authorization is only permitted to grant a positive one (pn = '+'), not a negative one (pn = '-').

Example 1 In an application system *Business_system*, if a registered user *Bob* pre-pays \$1000, he can enjoy a certain super-value service m for 6 times during every Friday since the time 09/12/2006. Let this privilege be *super*. This authorization can be expressed by a grant rule as the following:
access([09/12/2006,+ ∞] *, Weeks+5.days, (6, Bob , m, super, +, Business_system))*
← *prepay(Bob,1000)* & *register(Bob)*
Here *prepay(Bob, 1000)* means Bob pre-pays \$1000 and *register(Bob)* Bob is a registered user. Both of them are trigger condition expressions.

Definition 7 (Derived Rule) *A derived rule is defined as the form of:*
deraccess (time, period, auth) ← $L_1\& \ldots \& L_n$

where L_i can be *access with conditional expressions.*

Derived rules can be used to update usage times during ongoing usage control. When an access is performed, the usage times is decreased by 1. However, derived rules only update times authorizations rather than non-times authorizations after accessing, which is discussed in resolution rules,

Derived rules also support transferring times-based authorizations. In TUCON, transferring authorizations means consuming the usage times of privileges on digital objects. When the usage times of a privilege decreases to zero, the privilege is automatically revoked by the system.

Example 2 Now Bob wants to transfer 3 times for enjoying the service m to another user Alice. These can be defined with derived rules as the following:
deraccess([09/12/2006,+ ∞] *, Weeks+5.days, (3, Alice , m, super, +, Business_system))* ←*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (6, Bob , m, super, +, Business_system))* & *give(3, Alice, m, super, Bob)* & *less(3,6)*
deraccess([09/12/2006,+ ∞] *, Weaks+5.days, (3, Bob , m, super, +, Business_system))* ←*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (6, Bob , m, super, +, Business_system))* & *give(3, Alice, m, super, Bob)* & *less(3,6)*
where *give(3, Alice, m, super, Bob)* states that Bob transfers 3 times of privilege *super* to Alice. *less(m,n)* states true if m is less than n.

Through above discussion, we can notice that multiple times authorizations can be given for a subject to access the same object through applying grant and derived rules. After transferring authorizations or accessing, the usage times of this authorization can be decreased to zero. So we need a rule to resolve these conflicts. A resolution rule, given below, forces a final unambiguous decision to be made.

Definition 8 (Resolution Rule) *A resolution rule is defined as the form of: force_ access (time, period, auth)* $\leftarrow L_1\& \ldots \& L_n$

where L_i can be *access or deraccess or condition expressions*.

A resolution rule states that a given subject must be allowed/forbidden to perform a privilege on an object. Compared to grant and derived rules, which may cause authorizations conflicts, resolution rules state which authorizations the system must consider valid for each subject, on the basis of the existing granted or derived authorizations.

Resolution rules can be used to revoke authorizations, combine multiple times authorizations, update non-times authorizations after accessing and solve conflicts caused by times and non-times authorizations coexisting for a subject.

Example 3 In example 2, if Alice has 4 times *super* right on service *m*, a resolution rule should be used to solve conflicts after Bob transfers rights to Alice:
force_ access([09/12/2006,+ ∞] *, Weaks+5.days, (7, Alice , m, super, +, Business_system))* $\leftarrow$*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (4, Alice , m, super, +, Business_system))* & *deraccess (* [09/12/2006,+ ∞] *, Weeks+5.days, (3, Alice , m, super, +, Business_system))*
If Alice has non-times right on service m, this resolution rule should be used:
force_ access([09/12/2006,+ ∞] *, Weeks+5.days, (-1, Alice , m, super, +, Business_system))* $\leftarrow$*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (-1, Alice , m, super, +, Business_system))* & *deraccess (* [09/12/2006,+ ∞] *, Weeks+5.days, (3, Alice , m, super, +, Business_system))*
If Bob's has 0 times right after transferring rights, another resolution rule is added to revoke Bob's authorization:
force_ access([09/12/2006,+ ∞] *, Weeks+5.days, (0, Bob , m, super, -, Business_system))* $\leftarrow$*deraccess (* [09/12/2006,+ ∞] *, Weeks+5.days, (0, Bob , m, super, +, Business_system))* .

Depending on different situations, an administrator of the system can make a forced authorization decision by this rule. For example, when a security administrator notices that a user often sends many access requests without using services, this administrator may take actions on this user to prevent denial of service (DoS), such as revoking his authorization. Different applications have different considerations for administrators. However, as a general model, TUCON does not take any specific application into consideration. All these can be abstracted into condition expressions. Note that the conditions in resolution rules are factors which violate the security policy of systems, while those in grant rules are requirements, which must be satisfied for granting privileges.

Based on above given authorization rules, a set of rules are introduced to enforce the policy in TUCON. In the following, we just write the tuple of *auth* instead of the tuple of *(time, period , auth)*, based on the assumption stated in Section 4.1.

First, some symbols are explained to express rules as follows.

- *s, s1, system* $\in S$, where *system* is considered as the administrator and every user is a legal (registered) one in the system;
- *priv* $\in P$;
- $o \in O$;
- $Tr_i (i \in \mathbb{N})$ is the logic expression of a trigger;
- *give(s, s1, o, priv, pt)*: indicating that user *s* gives his *pt* usage times of privilege *priv* on object *o* to user *s1*;
- *accessing(s, o, priv)*: indicating that user *s* is performing operation privilege *priv* on object *o*;
- *expired(current_t)*: indicating that the current time is expired for the valid time interval, *current_ t* $\in T$.

R1 *auth(pt, s, o, priv, +, system)* $\leftarrow Tr_1 \& \ldots \& Tr_n \ (0 \leq i \leq n)$
R2a *auth(pt1, s1, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *give(s, s1, o, priv)* & *(pt* $\geq$ *pt1)*
R2b *auth(pt-pt1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *give(s, s1, o, priv)* & *(pt* $\geq$ *pt1)*
R3a *auth(pt-1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *accessing(s, o, prv)*& *pt > 0*
R3b *auth(-1, s, o, priv, +, system)* $\leftarrow$ *auth(-1, s, o, priv, +, system)* & *accessing (s, o, prv)*
R4 *auth(pt+pt1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *auth(pt1, s, o, priv, +, system)* & *(pt* $\geq$ *0)* & *(pt1* $\geq$ *0)*
R5 *auth(-1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *auth(-1, s, o, priv, +, system)*
R6 *auth(0, s, o, priv, -, system)* $\leftarrow$ *auth(0, s, o, priv, +, system)*
R7 *auth(pt, s, o, priv, -, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *expired(current_ t)*

- Rule R1, a grant rule, says that if user satisfies requirements Tr_i before allowing access, he/she can get *pt* times for operating privilege *priv* on object o from system.
- Rule R2a, R2b, two derived rules, implement that user can give his *pt1* usage times of privilege to user *s1*.
- Rule R3a, a derived rule, says that when user *s* with a times authorization is accessing object *o*, his usage times should be decreased by 1. Rule R3b, also a resolution rule, says that when user *s* with a non-times authorization is accessing object *o*, there is no need to update his authorization.
- Rule R4, a resolution rule, says that when there exist two times-based authorizations with the same user *s* for the same privilege *priv* on the same object, usage times should be added between these authorizations to make the final authorization decision. This rule is to solve authorizations from grant rules and derived authorization rules.

– Rule R5, a resolution rule, solves the conflicts between a *non-times authorization* and a *times authorization* for the same subject, object and privilege. When there exists the above case, a *non-times authorization* to the subject will be granted.
– Rule R6, a resolution rule, says that if usage times is zero, this authorization will be revoked.
– Rule R7, a resolution rule, says that if the valid time is expired, this authorization will be revoked.

4.4 Completeness

A set of the above rules can preserve the completeness property of the policy in TUCON.

THEOREM 1 (Completeness) *The policy in TUCON can be specified by a non-empty set of TUCON rules.*

Proof. If we can prove that there is no conflict decision by using these rules and these rules specify all possible decisions during the usage process in TUCON, the completeness of the policy in TUCON is guaranteed.

(1) no conflict decisions

By construction of these rules, resolution rules can be used to resolve the conflictions caused by grant rules and derived rules. After using resolution rules, there are no conflict access decisions since the resolution rule states that a subject must be allowed/forbidden to performance a privilege on an object. So we can safely conclude that there is no conflict decision made by using these rules.

(2) specifying all possible decisions

If the system state transitions in TUCON satisfy these rules, it can conclude that these rules specifies all possible decisions during the usage process.

Based on Figure 1 in Section 2, the state transitions are constructed with the following steps and illustrated in Figure 2.

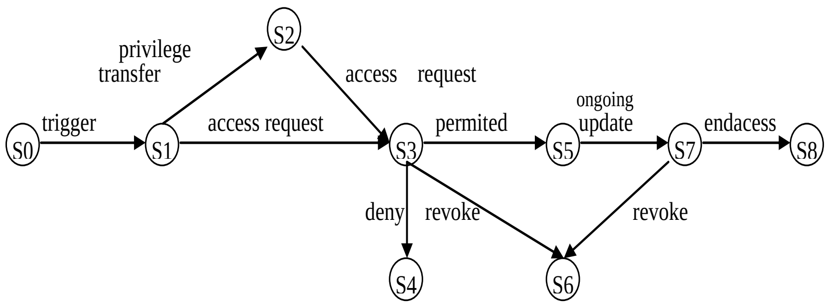

Fig. 2. State transitions.

1. Initially the system state is *S0*. In the state *S0*, the subject *s* performs some triggers satisfying with *R1*. Then *s* gets a new authorization and the system state changes to *S1*.
2. In *S1*, if the subject *s* transfers some privilege to another subject, the system state arrives *S2*. During changing of states, we can use *R2a and R2b* to transfer privileges. When causing some conflict authorizations, *R4* or *R5* can be used to resolve them.
3. In *S2* or *S1*, when receiving an access request, the system state changes to *S3*.
4. In *S3*, access requirements are satisfied, the access is permitted and the new system state is *S5*. If usage times is zero, *R6* is used to revoke this privilege and new system state is *S6*. If the valid time is expired, *R7* is used to revoke this privilege and new system state is *S6*; Otherwise, the access is denied and the system state changes to *S4*. Notice that in *S4*, the privilege is not revoked by system.
5. In *S5*, after using *R3a and R3b* to update authorizations, the system new state is *S7*.
6. In *S7*, if usage times is zero, we revoke the privilege with *R6* and arrive at the state *S6*. Otherwise, the system state is *S8* after ending an access.

With simple model checking, we can verify that all the rules are satisfied in these state transitions. That is, these rules specify all possible decisions in TUCON.

Considering the two above factors, the policy in TUCON can be specified by a non-empty set of the TUCON rules. □

5 Implementation of TUCON

In the above section, TUCON has been discussed in detail. Now, the implementation of TUCON in practice is given, which includes administration of authorizations and implementation of access control.

5.1 Administration of Authorizations

In the implementation of access control models, the most important thing is administration of authorizations. All authorizations in TUCON are derived from grant, derived, and resolution rules. A set of authorizations is called a Times-based Authorization Base (TAB). A TAB includes authorizations from access, deraccess, and force_ access which are not conflict with each other.

In a TAB, operations of authorizations are to grant/revoke times and non-times authorizations to/from users. In order to support these operations, the following problems must be solved:

(a) How to deal with a situation when times and non-times authorizations co-exist for a given object, subject, and privilege?

(b) How to deal with multiple times authorizations for a given object, subject, and privilege?

We can deal with the above problems easily with the above Rule *R4* and Rule *R5*.

Next we discuss operations for revoking privileges. It includes two kinds of manipulations: repeal non-times authorizations and automatically revoke times-based authorization. First, we check an authorization tuple $au = ([t_a, t_b],\ d,\ (pt,\ s,\ o,\ priv,\ pn,\ g))$ with respect to the current time. If the current time $current\_time > t_b(current\_time \in T)$ then set $pn =$ '-' in the 6-tuple of authorization; If $t_a \leq current\_time \leq t_b$, tuple au should be further checked to make sure whether it is a times-based authorization or not, and then if $\sigma_{au}(pt) = 0$ set $pn =$ '-' in the 6-tuple authorization *au*. Finally, we delete the tuples with $pn =$ '-' in TAB to form the new TAB. Otherwise, TAB will not be changed.

5.2 Access Control

After a subject is granted with an authorization, an access request is needed to access the object.

Definition 9 (Access Request) *An access request is a 5-tuple* (t, p, s, o, priv) *where $t \in T$ is the time when the access is requested, $p \in Q$ the current period point, $s \in S$ the user who requires the access, $o \in O$ the object to be access, and $priv \in P$ a privilege exercised on object* o*.*

As far as an authorization is concerned, the first step is to judge whether this authorization is valid or not. So every access request is checked against the current TAB to determine whether the access is authorized, which is checked by the valid authorization function.

Definition 10 (Valid Authorization Function) *The valid authorization function is used to judge whether the current authorization is valid. It can be expressed as the following:*

$$G(r) = \begin{cases} au & au \in TAB \wedge \sigma_r(t) \in \sigma_{au}(time) \\ & \wedge \sigma_{\sigma_{au}(auth)}(s, o, piv) = \sigma_r(s, o, priv) \\ \emptyset & others \end{cases}$$

where r is an access request. *G(r)* returns an authorization tuple. When it is $\emptyset$, the authorization is illegal, otherwise is a legal authorization.

However, a valid authorization is not enough for an access request. The specific period of the current request access should also be checked, which is valid or not according to the period constraint of an authorization. A valid authorized access request is a request for which an authorization exists in the current TAB, which is checked by the following valid access function.

In order to express the definition of a valid access function conveniently, a useful expression is given for the relationship between a period point and the period. If a specific period point $p \in Q$ belongs to the period $per \in Q$, it is denoted as $p = \prod(per)$.

Definition 11 (Valid Access Function) *The valid access function is used to judge whether the access request is valid according to the current TAB. It can be expressed as follows:*

$$F(r) = \begin{cases} true & \exists G(r)(\sigma_r(p) = \prod(\sigma_{G(r)}(period))) \\ false & others \end{cases}$$

where r is an access request. If *F(r)* is *true*, the access is valid.

After a subject submits an access request r and *F(r)* is *true*, this subject is permitted to access the object. During the following process of accessing, there are three kinds of situations that should be considered. If a requested authorization tuple is a non-times authorization, the TAB remains unchanged. If it is a times authorization, when the times is more than zero, the number of the privilege times is subtracted by 1, and then the TAB by resolution rules is modified. If the number of times is zero, we delete this tuple from the TAB. The concrete algorism is briefly described in the following, where the TAB is current times-based authorization base, and r is an access request.

Access_ control(TAB, r)

```
{......
 $au = G(r)$;         // use the valid authorization to return a set function of
                      // authorization tuple, and then judge whether the
                      //authorization is valid
 if ( $au = \emptyset$ )
           error("Illegal Authorization ");
 if ($\sigma_{au}(pt) = 0$) // judge whether the privilege times is 0, in order
                      // to decide whether the authorization should be revoked or not.
     { revoke the authorization set pn ='-' in the 6-tuple
       of authorization, and delete those derived authoriza-
       tion tuples by resolution rules with pn ='-1', and
      form the new TAB;
      error ("This authorization does not exist ");
     }
 $k = F( r )$;        // use the valid access function to return a
                       // boolean value, with which to judge
                      // whether the access is valid.
 if   ( $k = false$ )
          error ("Illegal Access ! ");
 $i = \sigma_{\sigma_{au}}(pt)$;      // the times of privilege.
  if ( $i > 0$)
    { get by subtracting 1 from au's privilege times,
      do some modifications in the TAB and then get
      the new TAB
    }
    ......
 }
```

6 Conclusion and Future Work

To meet new requirements in recent information systems, this paper presents a new access control model— the Times-based Usage Control (TUCON) Model, TUCON supports both consumable and temporal authorizations, and persistent authorizations with transferring authorizations. The key concept of TUCON is the usage times of privileges, which makes the implementation of access control more active and mutable, and protects resources from being abused.

TUCON has a vast range of applications in modern information society and can effectively solve the problems of consumed privileges in the validity of the period, especially in the times-metered systems and electronic commerce systems.

TUCON can be viewed as one of the specific research problems for mutable attributes [15] in modern access control. In order to clearly express our key point of this model, we analyze TUCON with different usage times, just assuming that the same privileges on the same digital objects have the same time interval and the same period. It is an interesting research topic to consider the different time intervals and different periods in TUCON. This research work is currently under our way. Based on the current progress of TUCON, development of the administration of authorizations in UCON is also future research work.

References

1. A. Gal, V. Atluri. An Authorization Model for temporal Data. *ACM Transactions on Information and System Security*, Volume 5 ,Issue 1,Feb. 2002.
2. B. W. Lampson. Protection. *5th Princeton Symposium on Information Science and Systems*, 1971. Reprinted in ACM Operating Systems Review 8(1): 18-24, 1974
3. C. Landwehr. Protection (Security) Models and Policy. *The Computer Science and Engineering Handbook* , CRC Press, pp 1914-1928, 1997
4. D. E. Bell ,and L. J. Lapadula . Secure computer systems: Unified exposition and Multics interpretation. *Technical Report ESD-TR-75-306*,The Mitre Corporation, Bedford, MA, March 1975.
5. D. E. Denning. A lattice Model of secure information flow. *Communications of ACM.* 19(5): 236-243. 1976
6. D. D. Downs, J. R. Rub, K. C, Kung, and C. S.Jordan. Issues in discretionary access control. *In the procceding of IEEE Symposium on Research in Security and Privacy*, pages 208-218, Apr 1985.
7. E. Bertino, C. Bettini, and P. Samarati. A Temporal Authorization Model. *CCS '94*,l/94 Fairfax Va, USA 1994
8. E. Bertino, C. Bettini, and P. Samarati. A Temporal Access Control Mechanism for Database Systems. *IEEE Transactions on Knowledge and DataEngineering*,Vol.8, No.1, Feburary 1996.
9. E. Bertino, C. Bettini,E. Ferrari, and P. Samarati. An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning. *ACM Transactionon Database Systems*,Vol.23, No.3, September 1998
10. E. Bertino, P.A Bonatti , and E. Ferrari. TRBAC: A Temporal Role-based Access Control Model. *ACM Transactionon on Information and System Security*,Vol.4, No.3, pp.191-233, Aug 2001.

11. F. Kargl, J. Maier,and M. Weber. Protecting Web Servers from Distributed Denial of Service Attacks. *Proceedings of WWW '10*,514-525, 2001
12. G. S. Graham, and P. J. Denning. Protection - Principles and Practice. *In Proceedings of the AFIPS Srping Joint Computer Conference*, volume 40, pages 417-429, AFIPS Press, May 16-18 1972.
13. James B.D. Joshi,E. Bertino, U. Latif , and A. Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IACM Transactionon on Knoledge and Data Engineering*,Vol.17, No.1, pp.4-23, Jan 2005.
14. J. Park, X. Zhang, and R. Sandhu. The Usage Control Model. *ACM Transactions on Information and Systems Security*, Feb. 2004
15. J. Park, X. Zhang, and R. Sandhu. Attribute Mutability in Usage Control. *IFIP WG 11.3*, Nov. 2004
16. Jams F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of ACM*,Volume 26, November 1983.
17. J. Lo. Denial of Service or "Nuke" Attacks. *http://www.irchelp.org/irchelp/nuke/*,Mar 12, 2005
18. M. Doerr, and A. Yiortsou Implementing a Temporal Datatype. *Technical Report ICS-FORTH/TR-236*, November 1998
19. M. Kudo, S. Hada. XML Document Security based on Provisional Authorization. *CCS'00*,Athens, Greece. ACM, 2000
20. N. Weaver. Warhol Worms: The Potential for Very Fast Internet Plagues. *http://www.cs.berkeley.edu/ nweaver/warhol.html*.
21. G. S. Griffiths, and B. W. Wade. An authorization mechanism for a relational database system. *ACM Transactions On Database Systems*.1(3):242-255, 1976.
22. R. Sandhu. Access Control: The Neglected Frontier (Keynote Lecture). *Australasian Conference on Information Security and Privacy*. 1996
23. R. Sandhu. Role Hierarchies and Constraints for Lattice-Based Access Controls. *European Symposium on Research in Security and Privacy*. 1996
24. R. Sandhu, and J. Park. Usage Control : A Vision for Next Generation Access Control. *The Second International Workshopon Mathematical Methods, Models and Architectures for Computer Networks Security*, 2003.
25. F. Siewe, A. Cau and H. Zedan. A Compositional Framework for Access Control Policies Enforcement. *In Proceeding of the ACM Workshop on Formal Methods in Security Engineering*. 2003.
26. S. Jajodia, P. Samarati, and V. S. Subrahmanian. A Logical Language for Expressing Authorizations. *IEEE Symposium On Research in Security and Privacy*, Oakland, California, 1997.
27. X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu A Logical Specification for Usage Control. *9th ACM Symposium on Access Control Models and Technologies (SACMAT)*.New York, June 2-4, 2004.
28. X. Zhang, F. Parisi-Presicce, J. Park, and R. Sandhu. Formal Model and Policy Specification of Usage Control. *ACM Transactions on Information and System Security (TISSEC)*. 8(4): 351-387, 2005.