Available online at www.sciencedirect.com

ScienceDirect

AASRI Procedia 5 (2013) 15 – 20

AASRI Procedia

www.elsevier.com/locate/procedia

2013 AASRI Conference on Parallel and Distributed Computing and Systems

A Location Temporal Based Access Control Model for IoTs

Chao Lee[a,b,]*, Yunchuan Guo[a], Lihua Yin[a]

[a]*Institute of Information Engineering, Chinese Academy of Science, Beijing,100185,China*
[b]*Graduate University of Chinese Academy of Science, Beijing,100049,China*

Abstract

Internet of things (IoTs) is a hot topic in recent years, and its various applications have been applied in several fields. The essence of IoTs is to connect all the things including the devices embedded RFID, sensors, etc. together, to help people utilizing the information from the things to improve their life. The things are always located at different places at different times, and the information of the thing may have different access right at different location during different time. Therefore, the access control mechanism in IoTs should consider location and temporary constraints in order to enhance the security of an application. In this paper, we propose a model that combines location and time with security level to control access to the information within the sensing layer of the IoTs. This model is called Location-Temporal Access Control Model(LTAC). The main principle of LTAC is its use of a lattice approach to decrease the size of policy bases. We show that how the different components in the model are related with location and how this location information can be used to determine whether a subject has access to a given object.

Keywords: Access Control; Internet of Things; Location; Temporal; Information policy

* Corresponding author.
*E-mail address:*lichao@nelmail.iie.ac.cn

1. Introduction

Internet of things (IoTs) is a hot topic in recent years, and its various applications have been applied in several fields[1] . The essence of IoTs is to connect all the things including the devices embedded RFID, sensors, etc. together, to help people utilizing the information from the things to improve their life. The things are always located at different places at different times, and the information of the thing may have different access right at different location during different time. Therefore, the access control mechanism[2] in IoTs should consider location and temporary constraints in order to enhance the security of an application.

The mainly existing access control schemes for the IoTs concentrate on authentication while ignoring more general models. To summarize, Benenson etc. provides a solution for defense against node capture through the collaboration of sensor nodes [3], while Shen proposes three access control schemes using the one-way key chain and the Merkle hash tree [4]. In [5], a solution to the problem of new node admissions is provided; however, the solution is not secure against replay attacks according to [6]. [7] lists influential works. These studies favor specialized schemes and show that few general access models are proposed for the IoTs.

In this paper, we propose a model that combines location and time with security level to control access to the information within the sensing layer of the IoTs. This model is called Location-Temporal Access Control Model(LTAC). The main principle of LTAC is its use of a lattice approach to decrease the size of policy bases. We show that how the different components in the model are related with location and how this location information can be used to determine whether a subject has access to a given object.

1.1. Location formalization

In the IoTs, there are several ways to represent location information, such as geometrical coordinate, symbolic representation, etc. In geometrical coordinate representation, a location is represented by 2 or 3 dimensional coordinate obtained by GPS devices, while in symbolic representation approach, each location is represented by a symbolic name, such as Beijing (a city name), CCTV building (a building name) and Room 656 (a room name).

Locations can be specified at different levels of granularity, and the smallest granularity of locations is a point. In this paper, we use the notion of logical location, which is formally defined as follows.

Definition 1. (Logical Location) A logical location $lloc_i$ is a non-empty set of points ($ploc_1,\ldots,ploc_n$). $LLOC=\{lloc_1,\ldots lloc_n\}$ represent a set of names of logical locations.

Given two logical locations, a containment relation may exist, so we define the location containment relation as follow.

Definition 2. (Containment relation) Logical location $lloc_i$ is contained in another logical location $lloc_j$, written as $lloc_i \subseteq lloc_j$, if $\forall ploc_i \in lloc_i, ploc_i \in lloc_j$.

1.2. Temporal formalization

In order to describe operations that can only be executed within a given period, we first model time. Here, we use the notion of a calendar [8]. A calendar consists of a countable set of contiguous intervals. In a typical example, a calendar is set in years, months, and days. Since two calendars can have different granularities, a sub-calendar relationship can be established among them: given two calendars c_1 and c_2, c_1 is a sub-calendar of c_2 (written as $c_1 \sqsubseteq c_2$) iff there exists $i \in N$ with $c_2 = i \times c_1$, where N is a set of natural numbers. For example, days are a representative sub-calendar of months. Obviously, $\sqsubseteq$ is a partial order relation. A calendar base, written as CB, represents a set of calendars and will generally change with different contexts.

For example, if a school curriculum is comprised of years, semesters, and weeks, then the corresponding *CB* is {years, semesters, weeks}.

Definition 3 (**Calendar time**): Given a *CB*={$c_1,\ldots,c_n$}, calendar time *ct* is defined as $ct=\sum_{i=1}^{n} n_i \cdot c_i$, where, $n_i \in N$ and for all $2 \le i \le n$ we have $c_i \sqsubseteq c_{i-1}$.

Let CT be a set of calendar times. Generally, any two calendar times are always comparable. That is, for any ct_x and ct_y in CT, we have $ct_x \le ct_y$, or $ct_y \le ct_x$. In the IoTs, there exist different types of time constraints such as the earliest access time, the latest access time, the earliest finish time, and the latest finish time, etc. In this section, the earliest access time (*eat*) and the latest finish time (*lft*) are adopted. Now, we formally define time constraints.

Definition 4 (**Time Constraints**): Time constraint $TC \subseteq CT \times CT$ is a set of two-dimensional vectors of calendar times, where the first dimension and the second dimension represent *eat* and *lft*, respectively. Time constraints must satisfy the following condition: for any $(ct_1, ct_2) \in TC$, $ct_1 \le ct_2$.

Example 1. Given TC={$(ct_{11},ct_{12}),(ct_{21},ct_{22})$}, and if an event satisfies *TC*, then (1) its starting access time is greater than or equal to ct_{11} and its finish time is less than or equal to ct_{12}; or (2) its starting access time is greater than or equal to ct_{21} and its finish time is less than or equal to ct_{22}.

Given a time constraint {(*1,3*),(*2,4*)}, the constraint can be reduced to {(*1*, *4*)}. This is related to the following definition.

Definition 5 (**Simplest Time Constraints**): A time constraint *TC* is the simplest, if for any (ct_{11},ct_{12}) and $(ct_{21},ct_{22}) \in TC$, min{ct_{12}, ct_{22}}<max{ct_{11}, ct_{21}}.

In the sequel, we assume that time constraints are always the simplest.

Definition 6 (**Order relation $\preceq$ on *TC***): Given any tc_1 and tc_2 in *TC*, $tc_1 \preceq tc_2$ (meaning that tc_2 is stricter than tc_1) iff $ct_{21} \le ct_{11}$ and $ct_{12} \le ct_{22}$, where $tc_1=(ct_{11},ct_{12})$ and $tc_2=(ct_{21},ct_{22})$.

Proposition 1. $\preceq$ on *TC* is a partial order.

Proof: Here we only prove that $\preceq$ is transitive. For any tc_1, tc_2 and tc_3 in *TC*, and let $tc_1 \preceq tc_2$, $tc_2 \preceq tc_3$, where $tc_1=(ct_{11},ct_{12})$, $tc_2=(ct_{21},ct_{22})$, $tc_3=(ct_{31},ct_{32})$. From $tc_1 \preceq tc_2$, we have $ct_{21} \le ct_{11}$ and $ct_{12} \le ct_{22}$; Similarly, from $tc_2 \preceq tc_3$, we have $ct_{31} \le ct_{21}$ and $ct_{22} \le ct_{32}$. Thus, $ct_{31} \le ct_{11}$ and $ct_{12} \le ct_{32}$. So, $tc_1 \preceq tc_3$.

In some cases, logical time (such as work time or class time) is more important. We define logical time as follows.

Definition 7. *timeAssigned:* $LT \rightarrow 2^{TC} / \{\varnothing\}$ is a function providing intersection from *LT* to the non-empty power set of *TC*, where $LT=\{lt_1,lt_2,\ldots,lt_n\}$ represents a set of names of logical time.

2. Location-Temporal Access Model

In this section, we discuss our formal model, LTAC.

2.1. Basic Components of LTAC

The basic LTAC model is comprised of the following components:

$ID=\{id_1,\ldots,id_n\}$ is a set of node IDs;

$LT=\{lt_1,\ldots,lt_n\}$ is a set of logical time constraints;

$LLOC=\{lloc_1,\ldots,lloc_n\}$ is a set of logical locations;

$OP=\{op_1,\ldots,op_n\}$ is a set of operations;

$O=\{o_1,\ldots,o_n\}$ is a set of objections;

$PERM \subseteq OP \times O$ is a set of permissions;

$ACCBASE \subseteq LT \times LLOC$ is a set of access bases;

$PA \subseteq PERM \times 2^{ACCBASE}$ is a many-to-many map of connections between permissions and the power set of access bases. $(x,y) \in PA$ means that any access base in y has permission x. For example, if $((op,o),\{(lt_1,lloc_1),(lt_2,lloc_2)\}) \in PA$, then a node satisfying the constraint of lt_1 at location $lloc_1$ can execute operation *op* to object *o*, and a node satisfying the constraint of lt_1 at location $lloc_2$ can execute operation *op* to object *o*;

$accbaseAssigned : PERM \rightarrow 2^{ACCBASE}$ assigns a permission level to access bases, where $accbaseAssigned(perm) = \{accbase \mid (accbase, perm) \in PA\}$;

$permAssigned : ACCBASE \rightarrow 2^{PERM}$ assigns an access base to permissions, where *permAssigned* (*accbase*)=$\{perm|(accbase,perm) \in PA\}$;

AccessRequest: ID×OP×O is a predicate; if this is true, then a node id requests operation op to object o;

AllowRequest: ID×OP×O is a predicate; if it's true, then a node id is allowed to execute operation op to o;

DenyRequest: ID×OP×O is a predicate; if it's true, then a node id is not allowed to execute op to o;

RevokeRequest: *ID×OP×O* is a predicate; if this is true, then the permission node *id* executes operation *op* to *o* will be revoked;

$CurrentTL : ID \rightarrow LT \times 2^{LLOC}$ maps *id* to access bases, which returns the current access time, and the current logical locations when node *id* requests an access.

In order to return the logical locations of a node, its physical location should be first obtained, and then its logical locations can be computed. Because a physical location can be assigned to many logical areas, the locations returned by *CurrentTL* comprise a set of logical locations.

2.2. Mechanism for authorization and revocation

Intuitively, if a node is located in an appropriate area with the ability to satisfy the given time constraints, its requests to execute some operations on an object should be allowed. In order to not use too many symbols, we overload the notion $\in$, as follows.

Given *id*, *op,* and *o*, let $CurrentTL(id)=\{lt_{id},\{lloc_{id1},\ldots,lloc_{idn}\}\}$ and $accbaseAssigned((op,o)) =\{(lt_1,lloc_1),\ldots,(lt_n,lloc_n)\}$, $CurrentTL(id) \in accbaseAssigned((op,o))$ iff there exists $(lt_i,lloc_i) \in accbaseAssigned((op,o))$ with $lt_{id} \in lt_i$ and $lloc_i \in \{lloc_{id1},\ldots,lloc_{idn}\}$. Now, we give the authorization schemes

(1) $AccessRequest(id,op,o) \wedge CurrentTL(id) \in accbaseAssigned((op,o)) \rightarrow AllowRequest(id,op,o)$.

(2) $AccessRequest(id,op,o) \wedge CurrentTL(id) \notin accbaseAssigned((op,o)) \rightarrow DenyRequest(id,op,o)$.

(3) $AllowRequest(id,op,o) \wedge CurrentTL(id) \notin accbaseAssigned((op,o)) \rightarrow RevokeRequest(id,op,o)$.

Formulas (1) and (2) show that: (1) if the a node *id* requests permission to execute operation *op* to object *o*, and *CurrentTL*(*id*) satisfies the conditions of executing *op* to *o*, then the requests will be allowed; (2) If a node id requests to execute operation *op* to object *o*, but *CurrentTL*(*id*) doesn't satisfy the conditions of executing *op* to *o*, then the request will be denied. Formula (3) suggests that if node *id* owns the permission of executing operation *op* to *o* and *CurrentTL*(*id*) is no longer able to satisfy the conditions of executing *op* to *o*, the permission will be revoked.

Figure 1 provides an overview of LTAC. This figure shows that even when an access request is allowed, it is still necessary to check this request with any time delay since reputation or location have the ability to change.

3. Access Lattice

Since PA is a many-to-many map of connections from permissions to access bases, the size of PA's access base may be large. Querying this map may consume more energy and computing resources, thus decreasing the efficiency of queries. If we assume that a node with a high reputation owns all the operations as a node having a lower reputation, then a corresponding access lattice constructed from access bases can be adopted to decrease the size of PA.

Definition 8 (*Order relation* $\leq$ *on ACCBASE*): Given any $(lt_1, lloc_1)$ and $(lt_2, lloc_2) \in ACCBASE$, $(lt_1, lloc_1) \leq (lt_2, lloc_2)$ iff $lt_1 \preceq lt_2$, $lloc_2 \subseteq lloc_1$.

Theorem 1: $(ACCBASE, \leq)$ is a lattice.

Now we give an example to show that using the lattice is an efficient way of decreasing the size of policy bases.

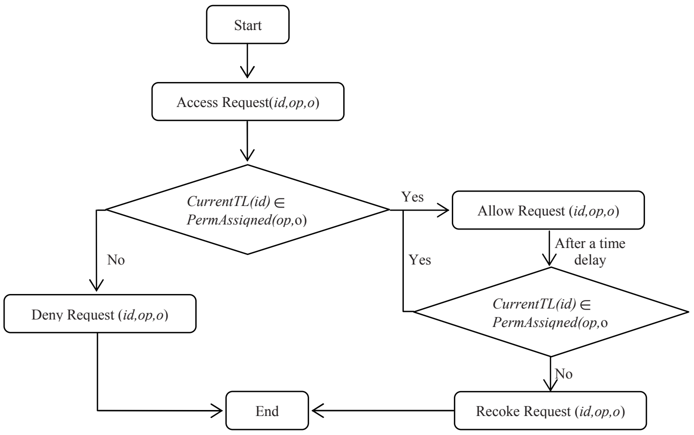
Start
Access Request(*id,op,o*)
CurrentTL(id) ∈ PermAssigned(op,o)
Yes
Allow Request (*id,op,o*)
After a time delay
Yes
No
Deny Request (*id,op,o*)
CurrentTL(id) ∈ PermAssigned(op,o
No
End
Recoke Request (*id,op,o*)

Fig 1 LTAC

Example 2. Let *PERM*={(*Open*,*MicroWave*), (*SetParameter*, *MicroWave*), (*close*, *MicroWave*)}, *ACCBSE*=$\{(lt_1, lloc_1), (lt_2, lloc_2), (lt_3, lloc_3), (lt_4, lloc_4)\}$. The access base for each permission is as follows: *accbaseAssigned*((*close*, *MicroWave*))= $\{(lt_1, lloc_1), (lt_2, lloc_2), (lt_3, lloc_3), (lt_4, lloc_4)\}$, *accbaseAssigned*((*SetParameter*, *MicroWave*))=$\{(lt_3, lloc_3), (lt_4, lloc_4)\}$ and *accbaseAssigned*((*Open*, *MicroWave*))=$\{(lt_4, lloc_4)\}$. From above, the cardinality of *accbaseAssigned*((*Close*, *MicroWave*)), *accbaseAssigned*((*SetParameter*, *MicroWave*)) and *accbaseAssigned*((*Open*, *MicroWave*)) is 4, 2, 1, respectively. Assuming that a lattice can be formed from these access bases as shown in Fig 2, then the access base for each permission could be changed as follows: *accbaseAssigned*((*close*, *MicroWave*))= $\{(lt_1, lloc_1)\}$, *accbaseAssigned*((*SetParameter*, *MicroWave*))=$\{(lt_3, lloc_3)\}$ and *accbaseAssigned*((*Open*, *MicroWave*))=$\{(lt_4, lloc_4)\}$. This means that the cardinality of all three bases is 1. Given an access request *AccessRequest* (*id*, *Close*, *MicroWave*) of node *id* locates at *lloc*$_3$ and the current time satisfies *lt*$_3$, then *AllowRequest*(*id*, *Close*, *MicroWave*) will be true since $(lt_1, lloc_1) \leq (lt_3, lloc_3)$. From this example, we are able to deduce that the size of policy bases can be decreased using an access lattice.

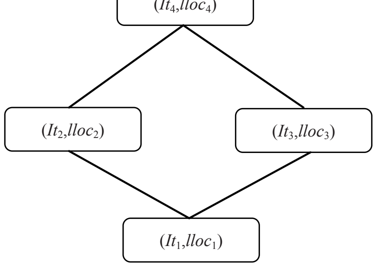

Fig 2. An Access Lattice

4. Conclusion

Traditional access control models do not take into account environmental factors before making access decisions. Such models may not be suitable for IoTs. Towards to this end, we have proposed a location-temporal access control model in this paper. In our model, the access to an object is determined by their location and time with reputation, to solve the problem of deciding *when* and *where* to authorize access requests, and *who* is able to access information, and utilize the notion of access lattice to decrease the size of policy base. In the future work, we will consider the situation that when a node moves into a location where communication is unstable, and the authorization information cannot be obtained in time.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (61070186) and the Strategic Priority Research Program of the Chinese Academy of Sciences(XDA06030200)

References

[1] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.
[2] D. Bell and L. La Padula, “Secure computer system: Unified exposition and multics interpretation,” Bedford, 1976.
[3] Z. Benenson, F. C. Gartner, and D. Kesdogan, “An algorithmic framework for robust access control in wireless sensor networks,” in Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on, 2005, pp. 158–165.
[4] S. Yu-long, M. A. Jian-feng, and P. E. I. Qing-qi, “An Access Control Scheme in Wireless Sensor Networks,” in Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on, 2007, pp. 362–367.
[5] H.-F. Huang, “A novel access control protocol for secure sensor networks,” Computer Standards & Interfaces, vol. 31, no. 2, pp. 272–276, 2009.
[6] I. Butun and R. Sankar, “A brief survey of access control in Wireless Sensor Networks,” in Consumer Communications and Networking Conference (CCNC), 2011 IEEE, 2011, pp. 1118–1119.
[7] J. Liu, Y. Xiao, and C. L. P. Chen, “Authentication and Access Control in the Internet of Things,” in Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on, 2012, pp. 588–592.
[8] J. B. Joshi, E. Bertino, U. Latif, and A. Ghafoor, “A generalized temporal role-based access control model,” IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 1, pp. 4–23, 2005.