A Flexible and Scalable Access Control for Ubiquitous Computing Environments[1]

Le Xuan Hung, Nguyen Ngoc Diep, Yonil Zhung, Sungyoung Lee, and Young-Koo Lee

Department of Computer Engineering, Kyung Hee University, Korea
{lxhung,nndiep,zhungs,sylee}@oslab.khu.ac.kr, yklee@khu.ac.kr

The *ubiquity* and *invisibility* characteristics of ubiquitous computing (*ubicomp*) arise many security problems, especially in the field of access control. Some important issues that are needed to be addressed in access control design are: (1) Ubicomp environment is composed of huge amount of entities. Therefore, determination of access rights must be based on *role* or *group of role*, instead of individuals. (2) The *context* (e.g. user's location, user's need, etc) changes dynamically over time. Hence authorization of user's accessibility is required to be based on such contextual information for proper enforcement of the required policies. (3) Users may not know what credentials are to be provided to access a specific service. In that case, the delivery service must support some interaction mechanism to explicitly acquire necessary credentials from users. (4) Numerous entities, roaming across different domains in ubiquitous network, are usually unknown to the local system. Access control should be based on the notion of trust to grant privileges in such circumstances. Lots of works have been done in the area of access control. Most of them have followed any of the four main approaches: Role-based Access Control (RBAC), Policy-based Access Control (PBAC), Context-based Access Control (CBAC), and Trust-based Access Control (TBAC). However, each of these approaches itself can not fulfill such security requirements of ubicomp. Hence we propose a Hybrid Access Control (HAC) model to tackle the problems of these approaches while taking their major advantages. HAC is hybrid of RBAC, PBAC, CBAC, and TBAC.

Architecture: HAC consists of *Trust Calculator (TC)*, *Context Provider (CP)*, *Policy Manager (PM)*, *Role Manager (RM)*, and *Access Control Manager (ACM)*. Trust Calculator is responsible for calculating trust value on principal P (user). Context Provider provides *user's context* (UC) and *system's context* (SC) to ACM. Policy Manager maintains *system policies* and *service policies*. System policy deals with the permission over resource access on the system level. Service policy is local policy of each service. It defines what actions can be performed on the services and who are authorized to perform them. Role Manager stores *roles* in hierarchical structure. Fig 1 depicts the workflow of HAC including three major operations: *deduction*, *abduction*, and *trust comparison*. When a user sends a service request (R_S) to the ACM along with his credentials (C_P), the ACM firstly perform *deduction*

[1] This work is financially supported by the Ministry of Education and Human Resources Development(MODE), the Ministry of Commerce, Industry and Energy(MOCIE), and the Ministry of Labor(MOLAB) through the fostering project of the Lab of Excellency.
Dr. Sungyoung Lee is the corresponding author.

operation (step 1, written in the circle). It evaluates the service request by using policy rules from Policy Manager and context (*UC, SC*) from Context Provider, and it makes a decision whether this request is permitted or not. If the service request is not allowed due to limited privileges of his credentials, the system will pass this request to *abduction* operation. By checking the request and system policies, the abduction operation finds the minimum additional credentials (C_m) that the user must provide in order to get access permission. This additional credential requirement is sent to the user (step 2). If the user provides such credentials, then there is no problem for him to access the resource (step 3). If the user is unknown to the system or he is not able to provide required credentials, Trust Calculator computes trust value on this principal (T_V) based on recommendations of other principals, history of its interaction with the system, and other factors. It then passes this value to Trust Comparison (step 5). If the trust value exceeds the predefined trust value of given service/resource, he will be permitted to access to that service/resource, otherwise the access is denied. After that, ACM sends a feedback to Trust Calculator to update interaction information (step 6).

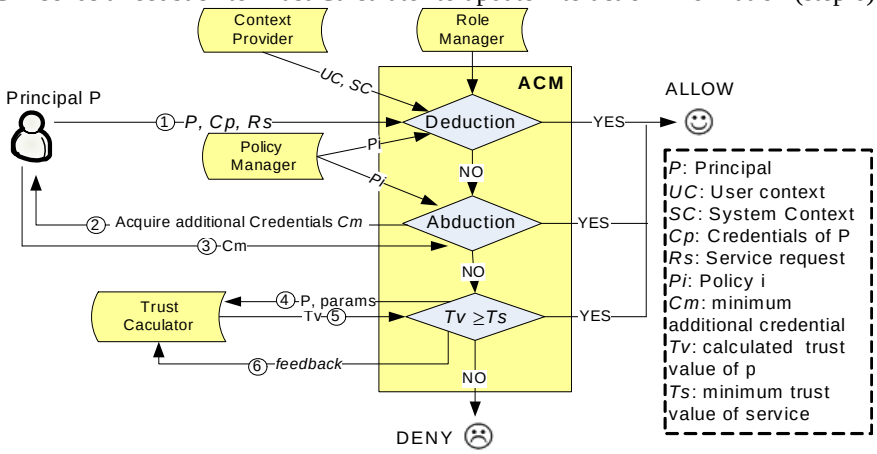

Fig. 1. Flow of Hybrid Access Control

By adopting role hierarchical structure, HAC can reduce the complexity and cost of security administration. HAC also uses context in design of principal to dynamically assign and adapt permissions to users. Whenever the context changes, role of user is also changed accordingly. By doing this, HAC provides flexible, convenient, and high secure accessing to resource. *Deduction* and *Abduction* are two fundamental operations that support users to specify the required credential for accessing the resource. These deal with the problem that a user usually does not have prior knowledge of required credentials to access certain resources. Moreover, trust management is involved as the final operation to deal with the problems of *uncertainty* between different domains. By applying trust, we want to provide as many services as possible to *unknown* users even though they can not show necessary credentials. Though HAC was introduced as a promising approach for ubicomp, it is not a complete security model. There are many works that we are still working on such as defining access control policy and integrating risk analysis.