5.5.2. Authentication for Ubiquitous Objects- an issue.

Ubiquitous computing technology provides an environment where users expect to access resources and services anytime and anywhere. The serious security risks and problems arise because resources can now be accessed by almost anyone with a mobile device in such an open model. This section explores security in ubiquitous computing with focus on authentication. We propose a new model, which uses distributed hierarchical tree-based approach for authenticating ubiquitous objects. We overcome the problems associated with having centralized source of information by dividing the information among Certifying Agents (CA), which are arranged in the form of a tree. Each CA maintains information about the local objects and its immediate descendant CAs. Certifying Agents communicate to each other for authenticating the ubicomp objects. The central idea of this solution is to efficiently spread the important information over several, separate Certifying Agents, which are distributed and hence highly available.

A Brief History:

A ubiquitous computing is a new computing paradigm, which integrates computation into our lifestyle and to extend our reach into a global network of computing, freeing us from desk-bound application interaction. With the ability to take corporate and personal processes and data with us, no matter our destination, opportunities abound for improving and enhancing our personal and professional life. The inherent freedom of ubiquitous system introduces challenges for security and trust management [73]. Arguably, trust management is the most critical security issue in ubicomp objects. If ubicomp objects do not have any prior knowledge of each other, the trust establishment becomes complicated. In these kinds of situations the ubicomp objects themselves should be responsible for their own security. But achieving this is impossible without depending on the third party for authentication.

Classical security models rely extensively on perimeter defenses and stable trust relationships. Thus the use of firewalls to enforce perimeter security based on a tightly defined network boundary. Also, users of a system are assumed to be pre-registered and
thus authentication and access control are centered on user identities. In other words, in 
centralized systems [89], authentication is completed via a login and password and 
transmitted through terminals directly connected to a host, and from which the access 
rights and authorization are based only on the user’s connection from the terminal and not 
on the validity or verification of the individual. In a pervasive environment, the above 
assumptions simply do not hold. Pervasive computing extends traditional computing 
boundaries. Also, trust-relationships are dynamic as the user community may be 
anonymous and constantly changing, making pre-registration unworkable, and the user-
identity may not be known, available or relevant [90]. The various security problems [88] 
that are barriers to mainstream pervasive computing are: device authentications, privacy, 
trust management, device assurance, resource, and availability. The trust, security, and 
privacy issues inherent in pervasive computing environments present unique challenges 
that require a fundamental re-examination of how to build large-scale, trustworthy, 
distributed systems. Ubiquitous systems need to be secure. The heart of ubiquitous 
computing vision lies in an inherent contradiction [75]. On the one hand, a computing 
environment must be highly knowledgeable about a user, to conform to his/her needs and 
desires without explicit interactions, almost reading the user’s mind. On the other hand a 
system that is truly ubiquitous will encompass numerous users, physical regions, and 
service providers. At such large scale, perfect trust among all parties is un-attainable and 
is ideal. Trust boundaries thus represent seams of discontinuity in the fabric of pervasive 
computing. Building secure systems is a challenge for the number of reasons. While 
establishing a trust [75], users must be confident of their computing environments, trust 
worthiness. The infrastructure must be confident of a user’s identity and authorization 
level before responding to the requests. This will become a key requirement as pervasive 
computing moves from the lab to the real world. The security issues for ubiquitous 
objects can be considered with respect to the following four criteria:

- **Confidentiality** is the guarantee that information is shared only between a user 
  and the entities the user is willing to communicate the information.
• **Authenticity** is the assurance that the ubiquitous object in a ubiquitous connection has the claimed identity and has subscribed to the ubiquitous service.

• **Integrity** means the correctness of stored and communicated personal data, in the sense, that only the corresponding person (the author or a responsible moderator) can alter them.

• **Availibility** means that the ubiquitous service is accessible and usable for subscribed persons using appropriate mobile devices.

**Authentication in Ubiquitous Computing:**

An authentication process establishes the identity of some entity under scrutiny. For example, a traveler authenticates himself to a border guard by presenting a passport. Possession of the passport and resemblance to the attached photograph is deemed a sufficient proof that the traveler is the identified person. The act of validating the passport (by checking a database of known passport serial numbers) and assessing the resemblance of the traveler is a form of authentication. Successful authentication does not imply that the authenticated entity is given access. An authorization process uses authentication, possibly with other information, to make decisions about whom to give access. For example, not all authenticated visitors will be permitted to enter anywhere in the defence building i.e., we need to establish some mechanism on, who can do what? and what not?, though they are authenticated visitors. For this reason we have introduced the concept of multi level security token concept. Each ubicomp object is issued a token at a particular level, depending on an associated risk involved by the third part Certifying Agents. When such object ‘A’, negotiate with other object ‘B’, it is up to the object ‘B’, to decide whether to permit ‘A’ for whatever it is requesting or not? depending on the security token level of the object ‘A’ and vice versa.

The existing security infrastructures [73] are inadequate for the increased flexibility required by distributed networks. We suggest, enhancing security by the addition of trust, which is similar to the way security is handled in human societies. A person is trusted if someone
we trust, says that the person can be trusted. In terms of distributed computing, a user is allowed to access a service or information, if the user has the access right to do so, or if the user has been delegated the ability by a trusted authority. Trust management can be viewed as developing of security policies, the assignment of credentials to entities, checking if the credentials fulfill the policy and the delegation of trust to the third party.

Related Work:

The five hard trust-related problems [88] in a pervasive computing environment are:

1. Who am I talking to?
2. Will my privacy be safeguarded?
3. Can I trust the device I am communicating?
4. Does the system provide the resource?
5. Will the pervasive services be reliably available?

At the present time, pervasive computing researchers are investigating specific security issues in the context of narrowly defined problems [88]. The main future challenge of pervasive computing is to offer access anywhere and anytime with any devices. However, before it becomes a reality, the problems of access control and authentication have to be solved, among others. Reijo Savola et al [73], proposes an approach for authentication using self-signed certificates. Here the node creates and signs a public key certificate using corresponding secret key. The self-signed certificate is not a proof of identity. But the proof that the node possess public secret key pair. The certificate gradually becomes proof of identity, when other nodes have signed it. The self signed certificate approach suits the situations in which the same node forms the network regularly. Here the first contact is insecure. But in the subsequent contacts, the nodes have sufficient information about each other, and hence the communication is secured.

The APC (Access Pass Certificate) proposed [82] has a model to enable an authorized user to roam and to access trusted hosts without being known locally. Each user can have an
APC certificate from two kinds of hosts: the main host (where the user is a member) and the trusted host (that trust the user). Using these certificates, the user extends progressively his access scope. Moreover, this model implements a decentralized mapping policy, where correspondence between the user's home profile and its rights on the trusted hosts is determined as needed. The main disadvantage of this model is the difficulty while managing relationship among organizations (hosts) and applying the mapping policy. In fact, an organization, having a trust relationship with other organizations, must validate and value relations manually and is not acceptable in case of true pervasive system. Security in ubiquitous computing environment such as pervasive or Ad-hoc, security models based on trust, (PGP pretty good privacy) [84], X509[83], [85], [84] and [87], are implemented. Almost all these models use the delegation concept to extend the access scope. They can use a certification [85], [84], or agents [87], which enable any, authorized entity, the right to delegate an access to certain resources. The delegation mechanism is considered to be efficient, but not quite sufficient to perform a broad access, because the user's scope is restricted only to environments where he could be locally known. Consequently, he can have an access if there is at least one entity that trusts him. In a pervasive computing environment, users have many devices that are used to initiate or answer remote service requests, such as obtaining real-time stock quotes, handling corporate e-mail, or accepting telephone calls. We envision that in the future, many applications will be distributed, running across many of a user's specialized pervasive devices rather than on a single system. In this case, a user needs the ability to "log into" the personal pervasive domain, which spans each of the pervasive devices representing this user. In addition, the pervasive devices belonging to the user's pervasive domain must be able to represent this user to external services. The paper [89] solves the problem of managing the authorization for pervasive devices belonging to a user's personal pervasive domain by introducing a central personal authorization gateway that accompanies the user and allows pervasive devices in the user's pervasive domain to be automatically configured and authorized. The pervasive authentication domain proposed [89] consists of a Personal Authentication Gateway (PAG) and a collection of pervasive devices. The Personal Authentication Gateway is transparent to external parties and
constitutes the security hub for the domain. A pervasive device can request its security configuration at boot-time from the gateway or the pervasive devices can refresh their security configuration on demand. The architecture of the Personal Authentication Gateway and pervasive devices that implement the pervasive authentication domain is illustrated in figure 5.30. As PAD model works on central repository like model, it may have a poor performance when large number of pervasive objects are involved and hence is not scalable.

Proposed Model:

The proposed model is termed as: **Distributed Multi Level Security Token-Based Authentication for Ubiquitous Objects (DMSA)**.
DMSA operates in distributed, multi-level security environment, which is modelled as forest of hosts operating as Certifying Agents (CA). Certifying Agents authenticate the ubiquitous objects (O) for communication. CAs are arranged in the form of a tree, root being the single point authority, yet the whole structure is geographically distributed. Each CA maintains information about its immediate descendants, which can be CAs or ubiquitous objects, registered with it, as shown in the figure-5.31.

Figure-5.31: Tree Structure for Distributed CA.

To illustrate the working of the proposed model, consider the Certifying Agents in the hierarchy as shown in the figure-5.31. The authentication process is carried out in many phases as discussed below:
1. Registration Phase:

1.1 Registering Certifying Agent:

All the Certifying Agents which are responsible for registering objects in them need to register themselves with another Certifying Agent (parent).

1.2 Registering Objects:

The Ubiquitous objects which want to communicate with other objects need to register them by providing Object ID to any CA and get appropriate Security Token, depending on the level at which CA believe this object.

2. Authentication phase:

The communication between two Ubicomp objects commences with one object sending the service request message to the other. This initiates the authentication phase.

For illustration, consider two ubicomp objects: a Mobile handset and a Laptop. In the registration phase, the Mobile is registered with the Certifying Agent CA1.1.1.0 and the Laptop is registered with the Certifying Agent CA1.3.2.0. Assume that in some context, at some different place, a Mobile (initiator) wishes to access the service available at Laptop (Listener) and hence sends a request message to Laptop. Laptop forwards the message to its certifying agent CA1.3.2.0. The authentication process that is followed is as described below:

Scenario 1:

Here, each certifying agent stores the information about all other certifying agents of the distributed tree. The Mobile, registered at Certifying Agent CA1.1.1.0 sends a service request to the Laptop. The Laptop authenticates the Mobile by sending a message to its Certifying Agent CA1.3.2.0. Since each certifying agent has the details about all other
Certifying Agents, the CA1.3.2.0, forwards the authentication message request to CA1.1.1.0 directly without going through up or down the tree. The reply from the CA1.1.1.0 will be forwarded to Laptop by CA1.3.2.0. Depending upon the reply, the Laptop can provide access to Mobile. This scenario is illustrated in figure-5.32.

1. Access Request

2. Authenticate

CA 1.3.2.0

Information
Of All
CAs in Tree
Available

3. Reply

CA 1.1.1.0

Registered at CA 1.1.1.0

Registered at CA 1.3.2.0

4. Allow or Deny

Figure-5.32: Authentication Process for Scenario-1.

The problem with this scenario is, regularly advertising the status of CA, addition/deletion of CAs at any place, to all the CAs in the tree. One such problem is highlighted in figure-5.33, where CA1.1.1.0 is blocked by its parent CA1.1.0. By the time ‘t’, this information spreads to all CAs in tree, may be some CA say 1.3.2.0 may send message to blocked CA1.1.1.0 for authentication of some objects. This happens because the information sent by CA 1.1.0 about blocked CA1.1.1.is not reached CA 1.3.2.0. Now CA 1.1.1.0 may respond and hence violate the whole objectives defined.
Scenario 2:
In this scenario each Certifying Agent stores the information about its children CAs and about its parent CA only. Every CA is assigned a unique hierarchical address (referred here as Ubiquitous Address similar to Internet Protocol Address) by its parent Certifying Agent. The Mobile, registered at Certifying Agent CA1.1.1.0, sends service request to the Laptop. The Laptop authenticates Mobile by sending the message to CA1.3.2.0. Now CA1.3.2.0 searches CA1.1.0 in its table. Since CA1.1.0, is not registered under CA1.3.2.0, it forwards the message to its parent Certifying Agent or to its descendent Certifying Agents, by resolving the hierarchical address. Upon receiving the message from
CA1.3.2.0, the Certifying Agent CA1.3.0 also follows the same procedure and the message will be forwarded finally to CA1.1.1.0. The CA1.1.1.0 authenticates the Mobile and the reply will be sent to CA1.3.2.0 without going through up or down the tree. Now CA1.3.2.0 forwards this reply to Laptop. The scenario-2 is represented in figure-5.34. The drawback of this scenario is the time ‘t1’ required to forward the authentication message through the up and down the hierarchy. However from the observation it is clear that, this time ‘t1’ is more acceptable than the time ‘t’ mentioned in scenario 1.

**Data structures used:**

Every CA maintains the records in the form of tables. Each CA has two tables, one for the information about its descendent CA and the other for the information about the registered ubiquitous objects in that region. It also has static information about the parent
CA. The structure of the table to maintain the descendent CA’s information is as shown in figure-5.35 and for registered objects in figure-5.36.

<table>
<thead>
<tr>
<th>Descendant CA ID</th>
<th>Ubiquitous Address (UA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure-5.35: Table to Maintain Descendent CA’s Information

<table>
<thead>
<tr>
<th>Object-ID</th>
<th>Ubiquitous Address (UA)</th>
<th>Retention Period</th>
<th>Security Token (ST) Issued</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure-5.36: Table to Maintain Registered Object’s Information

Object ID : Unique object ID.

UA : Ubiquitous address for the objects and is different from Object ID.

Retention period : Maximum duration for which the entry will be maintained by the CA

Security Token(ST) : Value in this field identifies the access right for the object.

Every Object should register with the local CA by specifying its ID and other details required so that CA can believe it. Based on the given information security token will be assigned to the object for some duration. More freedom implies less retention period and vice versa. When the object wants to communicate with other objects it has to send a request message as shown in figure-5.37, to the object with which it wants to communicate.
Figure-5.37: Request Format.

CA : The address of the CA with which the object is registered.

ST : The values can be:

0 - No access (default)
1 - Read only
2 - Read and Write

These values are used for illustration purpose only.

TO : Object ID with whom my-object wish to negotiate.

The receiving object ( RO ) forwards this message to the local, it’s CA ( LCA ). If the sending object has registered with the local LCA then ST will be verified locally else the LCA will query its parent CA by sending the message as shown in figure-5.38.

Figure-5.38: Message Format for Query Between CAs

Finally, the message will be forwarded to the proper CA (SCA), in which the My-object whose authentication details are registered. After verifying, SCA will respond back to the requesting CA(LCA and finally to the object ( RO ), which can then provide service depending on reply received.
The proposed model was implemented using Aglet, an agent platform developed by IBM. The empirical results show that the performance is acceptable though it involves exchange of a huge message, which is essential; otherwise it is practically impossible to authenticate devices in pervasive world. The GUI of the implementation is shown in figure-5.39.

**Work Conclusion:**

Trends in pervasive computing are increasing the diversity and heterogeneity of networked objects and their constituent devices. Developing security protocols that can handle diverse and mobile devices, networked in various ways represents a major challenge. In this paper, we have taken a first step towards meeting one of such challenge i.e. authentication of ubiquitous objects. Empirical results have shown that the performance of the proposed DMSA model is acceptable in reality of pervasive world. It is also clear that perfect trust among all parties in pervasive world is un-attainable ideal without compromising the true autonomy i.e. we need to depend on the trusted third party for authentication.

*This work has been accepted as WiP in IEEE PERVERSIVE COMPUTING 2006.*
Figure-5.39: GUI of the System Implemented.