International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:12, No:6, 2018

Trust Management for an Authentication System in Ubiquitous Computing

Malika Yaici, Anis Oussayah, Mohamed Ahmed Takerrabet

***Abstract*—**Security of context-aware ubiquitous systems is paramount, and authentication plays an important aspect in cloud computing and ubiquitous computing. Trust management has been identified as vital component for establishing and maintaining successful relational exchanges between trading partners in cloud and ubiquitous systems. Establishing trust is the way to build good relationship with both client and provider which positive activates will increase trust level, otherwise destroy trust immediately. We propose a new context-aware authentication system using a trust management system between client and server, and between servers, a trust which induces partnership, thus to a close cooperation between these servers. We defined the rules (algorithms), as well as the formulas to manage and calculate the trusting degrees depending on context, in order to uniquely authenticate a user, thus a single sign-on, and to provide him better services.

***Keywords*—**Ubiquitous computing, context-awareness, authentication, trust management.

I. Introduction

COMPUTERS in general and data networks in particular have evolved so fast that they have become essential in a few years. They, thus, occupy an important part of the human daily life, and are to be more and more present daily. Indeed, they intervene in all areas, and are solicited continuously. They are ubiquitous in providing different services in order to facilitate and simplify tasks that should initially require more time and effort. This is what Mark Weiser [1] had imagined in his paradigm of ubiquitous computing for more than twenty years now. Indeed, it is he who is the father of the ubiquitous computing and who gave it that name and who imagined the role it could play.

The concept of ubiquitous computing has long remained in the laboratory boxes, first under this name, then under the name of contextual informatics. Also, like all computer networks, ubiquitous networks are subject to attacks and threats that compromise the proper functioning of the entire system. As a result, their security becomes paramount and more than obvious. Currently, a lot of work goes in this direction, which is why our work focuses on securing these context-sensitive systems, and more precisely on authentication, by introducing the notion of trust.

In this paper, we have developed a system that manages authentication in a context-aware ubiquitous system based on the notion of trust.

After this introduction, Section II deals with important works on trust management in cloud and ubiquitous computing. In Section III, Single Sign On mechanism and Trust definitions are detailed. Section IV presents the main contribution of this paper. Discussion of the results and a validation of the proposed solutions are given in Section V. A conclusion and future work finish the paper.

M. Yaici is with the Laboratoire LTII, University of Bejaia, Algeria (e-mail: yaici_m@hotmail.com).
A. Oussayah and M. A. Takerrabet are with Computer Department, University of Bejaia, Algeria.

II. Related Works

Context-aware authentication systems have been proposed in many research works.

The authors of [2] propose to enhance the Multi-factor Authentication based on Multimodal Biometrics (MFA-MB) for Cloud SaaS/PaaS scheme in order to improve the User eXperience (UX). A function for UX measurement is defined and a new algorithm based on a classification approach called Class-Association Rules (CARs) which integrate classification and association rules is proposed. These may allow guaranteeing an improved UX in the Cloud authentication process, considering the variables (Time, Place, Device, etc.) which govern the Users' authentication habits.

A secure biometric-based authentication scheme for multi-server environment using smart card, where all the servers need not to be trusted and the trusted third party is no longer required in authentication, is proposed in [3]. The proposed scheme supports two way authentication and session key agreement without control server involvement.

In [4] the location-based risk is measured for mobile authentication. The location is detected based on GSM cell IDs and Wi-Fi access point MAC addresses, then safe and non-safe locations are classified depending on time and date. A safe location (home) does not need any authentication, but a non safe location may require a more performing scheme (biometrics). Users can have control on these registered locations as well as the system. The same approach is given in [5] where a location dependent disclosure risk estimation is used as a decision support mechanism (a) for persistent authentication of users and (b) to make reliable authorizations to facilitate secure data utilization in pervasive computing applications. Depending on the measured risk value authentication may be needed or not. Affiliations based multiplicative attribute graph model is used as a computational tool to evaluate the location-dependent disclosure risk variations when the users change locations overtime and the location dependent disclosure risk estimates is derived using search theory and entropy.

The paper [6] is a survey on geo-location authentication techniques. Geodetic location authentication is continuous, cannot be hijacked and can be considered as an electronic

notary. It provides context-aware services that associate locations with user's identities and content. The techniques studied in this paper are on cyber-locator technology, location-based authentication with active infrastructure and through multiple authentication techniques, location and authentication based encryption scheme and finally proximity-based authentication.

Trust has been widely used in authentication. In [7] a review on existing solutions for trusted mobile computing is given. There are two classes of solutions: Hardware-based solutions which consists on secure elements (smartcards), trusted platform modules (controller and cryptography) and trusted execution environment (Operating systems like SierraTEE, Genode, etc.); and Software-based solutions which consists on coexisting virtualization environments (Virtual operating systems) on a Smartphone. A secure environment is defined as an isolated execution, a secure storage and a secure communications (provisioning).

Definitions of trust, trustworthiness, authentication and trustworthy authentication in ubiquitous and pervasive computing are given in [8]. Trustworthy authentication is not establishing trust between the authentication entities, but means that an entity not only authenticates him to another entity, but also assures the other entity that he is trustworthy in the following communications. The proposition given in the paper would be a two-level trustworthiness certificate associated to servers.

In [9] trust and reputation is used to qualify cloud services using enhanced mutual trusted and reputed access control algorithms. Trust is estimated based on user's behavior (history of communications and friends recommendations) and the cloud's service provider's reputation. The access control is granted if the user's trust value is above a trust threshold associated to each service and the user has a sorted list of services depending on their reputation.

In [10], the first contribution is identifying a set of categories for credentials and adapts them to the cloud context. The categories are useful to define what kind of information has to be represented in terms of credentials for a specific system or service, and how this information can be grouped and organized. The second contribution is the identification of important elements that have to be considered when adopting or developing a solution for authentication and authorization.

A trust model that is used to evaluate cloud service security strength is proposed in [11]. It consists of a trust value which can be evaluated by a list of parameters that covers almost all relevant aspects of security. These parameters are: identity management, authentication authorization, data protection, confidentiality, communication, isolation (avoid security breaks and restrict user access area), virtualization, and compliance (cloud service grated by an external authority). Cloud service features and specifications are used to evaluate the trust value and termed as static trust. Value of trust is affected, based on user experience and transactions over a period of time. A refined set of parameters are formulated to evaluate the trust dynamically. Static and dynamic trust altogether determines security of the cloud services.

Physical trust is also studied in [12], where a context of physical trust relationship is built between users by visual contact. The i-contact is to visually confirm the user's identity by the surrounding user's eyes, and the k-contact, dynamically, changes the authentication level using information collected through i-contact. The method is mainly used to unlock a mobile device and use resources accessed by the mobile.

Applications where context-aware authentication and security systems based on trust have also been proposed.

In [13] Vehicular Adhoc Networks (Vanets) are considered and their challenges. A VANET is a network where each node represents a vehicle equipped with wireless communication technology. This type of network can improve road safety, traffic efficiency, and many other traffic-related applications, minimizing their environmental impact and maximizing the benefits of road users. A cloud based security and privacy-aware information dissemination system through ciphertext-policy attribute-based encryption for access mechanism with batch verification is proposed. The proposed approach relies on a traffic management Bureau to manage the key generation, and a trusted authority for credentials management using bilinear mapping is proposed to detect malicious vehicles.

The authors of [14] proposed a multilevel authentication methods for Vanets based on trust. To access internet, the vehicles are authenticated using a direct trust evaluation based on a historical security records from an authority unit, and to communicate between vehicles, in order to add or delete a node, an indirect trust evaluation based on nodes recommendations is used. A method to detect malicious nodes is also proposed.

III. Preliminaries

Nowadays, the use of distributed systems is almost universal in all areas. That's why the security of these systems is crucial. For example, a successful attack on large-scale distributed systems gives a hacker access to thousands of machines in one go. More importantly, distributed systems often have privileged access between them, and after successfully getting into one of them, a hacker will be able to bounce back to other networks.

A. Authentication

Authentication is the procedure that consists, for a computer system, to verify the identity of an entity (person, computer ...), in order to allow the access of this entity to resources (systems, networks, applications, etc.). Authentication thus validates the authenticity of the entity in question.

There are four classic authentication factors that can be used to confirm the identity of a principal [8]:

- Use information that only the principal knows (password, personal identification number).
- Use unique information that only the principal has (birth certificate, identity card, identity card, smart card, property rights, electronic certificate, diploma, passport, health card, mobile phone, etc.).
- Use information that characterizes the client in a given context (photo, physical characteristics, fingerprints, iris recognition, etc.).

- Use information that only the principal can produce (signature).

Different authentication systems exist, among them we distinguish the Single Sign-On (SSO).

B. Single Sign-On

Single sign-on (or unique identification) is a method that allows a user to perform only one authentication to access multiple computer applications (or secure web sites). It reduces the risk of centrally managing user identities by an administrator, increases user mobility and productivity. This doesn't mean that the SSO system unifies account information for all services, applications and systems, rather it hides such a multiplicity of account information into a single account that the user needs to login [15]. Once the user login, the SSO system generates authentication information accepted by the various applications and systems. The implementation of a single account (login / password) for each user is a progress that should become widespread in the coming years to ensure better services to users of different applications.

A classification of SSO techniques is given in [15].

There are three main approaches for implementing single sign-on systems: the centralized approach, the federative approach, and the cooperative approach [16].

- Centralized approach: The basic principle here is to have a global and centralized directory or database of all users. It also helps to centralize the management of the security policy. This approach is mainly intended for services all dependent on the same entity, for example within the middleware management of a company.
- Federative approach: In this approach, each service manages a part of the user's data (the user can therefore have several accounts), but shares the information he has on the user with the partner services. This approach has been developed to respond to a need for decentralized user management, where each partner service wishes to maintain control of its own security policy, such as a set of commercial and organizational independent shopping sites.
- Cooperative approach: The cooperative approach assumes that each user depends on one of the partner entities. Thus, when seeking access to a network service, the user is authenticated by the partner on whom he depends. As in the federative approach, however, each network service independently manages its own security policy.

C. Trust

Trust is a basic fact of human life that does not know a consensual definition. Its introduction in computer science was intended to facilitate authentication between machines when they are required to often communicate so that they do not authenticate at each connection.

The great interest in trust is the guarantee of fault and intrusions tolerance, and to avoid classical security deficiencies. Trust is typically used to enrich exchanges; an entity may decide to interact only with others it trusts.

Cryptography and firewalls are examples of hard security mechanisms; their general properties are to allow full access or no access at all, so it assumes complete certainty. But the notion of trust implies some uncertainty, so an alternative of "soft security" will be more appropriate to the trust management. Flexible security is the term used by Rasmussen et al. [17] to describe the social control model that recognizes that malicious agents can exist among benevolent ones, and tries to recognize them. Trusted authorities are not an adequate approach for a large distributed system. Trust as defined by some designers of trusted systems is misleading because absolute trust does not exist. For that a dynamic trust management is necessary.

A trust relationship between A and B exists when A believes in the reliability of B, however the inverse relationship may not exist. characteristics of a trust relationship are the following [18]:

- Relativity: Absolute trust does not exist, it is relative. Thus, one server can trust another server in one context and be suspicious of it in another.
- Asymmetry: A server A can trust a server B, but the opposite is not necessarily true.
- Transitivity: A server trusts a server B, and the server B trusts a server C, so A can trust the server C. However, this relation is to handle with care not to fall on errors of trust level estimation.
- Arity: The trust relationship can be between two entities one to one, one to many, several to one or several to many.
- Dynamism: Trust increases or decreases depending on experience (direct interaction). It also evolves with time.

Memory is an important factor in the area of trust, different question may arise: should trust be accorded taking into account all previous interactions? Or should it only take into account the last n interactions? Some models have established a threshold that takes into account only the last n interactions while others have introduced a forgetting factor.

The concept of cooperation is linked to the concept of trust, to cooperate with someone we need to trust him, to know that he will not betray us or that he has the necessary skills.

Some models uses a cooperation threshold; to cooperate with an agent, trust in this agent in a particular situation must reach that threshold of cooperation.

Other models base their assessment of the trust degree on the fact that two agents can cooperate, compete or be in transaction. Thus we can deduce that the concepts of cooperation, reciprocity and reputation are interrelated.

IV. Proposition of a Context-Aware Authentication System

Our proposition concerns authentication and more specifically the Single-Sign-On. In order to adapt this model to context-aware systems, we propose a fusion of federative and cooperative approaches and introduce the notion of cooperation between servers. A user would depend on a server, to have access to a service provided by another server, it would be enough that he authenticates to the server on

which it depends provided that the two servers are partners. This concept of partnership should be based on trust. Thus, the trust of one server towards another would depend on several factors such as the degree of security of the server, the security policy used by the server, etc. The servers in question communicate with each other securely and transparently with the client.

A. Modelling

In what follows, we will define the assumptions and the terminology used in our proposal.

Terminology

- SPi: set of servers partners of server i.
- IdS: an integer to uniquely identify the server.
- IdC: an integer to identify the client from the server on which it depends.
- TL_{Si-Sj}: the trust level of server i in server j. It is a real number in the range $]0, 1[$.
- TL_{Si-Cj}: the trust level of server i in the client j. This level is stored at the server i permanently but its value is, nevertheless, variable. It is a real number in the range $]0, 1[$.
- CTL_{Si-Cj}: the current trust level of the server i in the client j. This trust level is calculated at each session and varies primarily according to context-related parameters. It is a real number in the interval $]0, 1[$.
- TT_{Si}: Each server i has a minimum trust threshold TT_{Si} from which it becomes a partner with another server j if TL_{Si-Sj} is greater than or equal to TT_{Si}. It is a real number in the interval $]0, 1[$.
- TT_{Si-sj}: Each server i has a minimum trust threshold TT_{Si-sj} for a service j from which it will or will not provide the service to the client if TL_{Si-Cj} is greater than or equal to TT_{Si-sj}. It is a real number in the interval $]0, 1[$.
- KPRCA: the client-side private key of the server A.
- KPBCA: the client-side public key of the server A.
- KPRSA: the server-side private key of the server A.
- KPBSA: the server-side public key of the server A.
- KPRC: the private key of client C.
- KPBC: the public key of client C.
- Ks: the session key shared between two entities. This key is used to encrypt communications for a single session.
- Ts: the time duration of a session.
- Iv: the internal clock instant value of the client C.
- login/psw: the client login information and password.

Hypotheses

- Since the client depends on a server, it is assumed that the client has full trust in this server.
- We assume that the minimum level of a TL_{Si-Sj} is equal to 0.1.
- If the trust level TL_{Si-Sj} is zero, the server j is malicious. And if the trust level TL_{Si-Cj} is zero then the client j is malicious.
- Trust networks are established beforehand.

1) Description of Messages:

1) Authentication-request: Auth-request(((H (login, psw))KPRC, idC) KPBCi): This is the message sent by a client C to the server i in order to be authenticated.
2) Authentication-response: Auth-response((TL_{SA-CC}, Ks, idC)KPBSA, (Ks)KPBC): This is the response of the server A for the authentication request of a client C.
3) Service-request((Ks, Ts, idC, TL_{SA-CC}) KPBSi, (idC, Iv) Ks): This is the message sent by the client C to the server i in order to access a service
4) Service-response(Bool): This is the response of the server to the client C service request. The response is represented by a Boolean variable that is true if the service is granted to the client, or false otherwise.

B. Operation

First we will assume the following:

- A Server A has a pair of private key / public key (KPRCA and KPBCA). This pair of keys is used to encrypt a communication between the client and this server. The public key is known by the client.
- A Server A also has another pair of private / public keys (KPRCA and KPBSA). This pair of keys is used to encrypt communications between the server and its partners (the servers that belong to SPA). The public key is only known by these, and is not known by users.
- A server A has in his databases user's accounts (login and password hashes, and other personal information) that depend on him and trust levels TL_{SA-Cj} assigned to them. These are not known to users.
- Each client also has a couple of private / public key (KPRC and KPBC) to encrypt communications between the client and the different servers.

In the following scenarios, we will consider the client C, the server A on which the client C depends, the server B which is a partner server of A, the server D which is partner with B but not with A.

1) Scenario 1. Where Server A Is Nearest: A client C wants to authenticate to a server A on which it depends as shown on Fig. 1. To do so, it sends an authentication request to A containing its hashed login and password (using hash function MD5 [19] for example), all encrypted with its private key KPRC. Then, add its identifier to the message and then encrypt the whole by the public key of A KPBCA.

When the server receives the request (Algorithm 1), it decrypts it with its private key KPRCA, then with the user's public key. He compares the hash of the login and the password with those stored in his database if they correspond. If this is the case, the server responds to the client with a positive response, a session key Ks, a session validity Ts and its identifier and its trust level TL_{SA-CC} encrypted with the server-side public key KPBSB, plus the session key Ks encrypted with the client's public key KPBC.

The client wants to access a service provided by the server B partner of A, so he sends the session key Ks, the session validity Ts, its identifier and its trust level TL_{SA-CC}

encrypted with the sever-side public key of the server (as sent by the server A) plus the client identifier and the clock all encrypted with the session key Ks it has recovered. Then the server B recovers the trust level TL_{SA-CC}. Then, calculate the trust level TL_{SB-CC} using (1).

$$TL_{SB-CC} = TL_{SB-SA} * TL_{SA-CC} \quad (1)$$

TL_{SB-SA} being the trust level of the server B towards the server A. It retrieves the session key Ks with which it decrypts the second message to thus recover the identifier of the client as well as the clock instant value Iv to compare them with the identifier and the session validity Ts in order to verify respectively the identity of the client and the possible expiration of the session. Once these tests are validated, the server B checks whether or not the trust level TL_{SB-CC} allows the user to access the service (Algorithm 2).

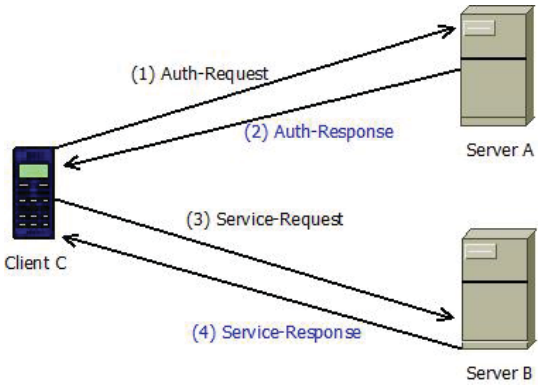

Fig. 1 Scenario 1 authentication diagram

Algorithm 1 Request-Authentication processing algorithm:

When server A receives an authentication request from a C client:
`Receive Auth-request ((H(login, psw)) KPRC, idC) KPBCA;`
`Extract (idC, H(login, psw)) KPRC;`
`Verifier (idC);`
`Recover (KPRC);` //from its database
`Recover (H(login, psw));` // from the received message
if `((H(login) == stored login) and (psw == stored psw))` **then**
  `Generate (Ks, Ts);` // Generate the session key and its time-out
  `Send ((`TL_{SA-CC}`, Ks, Ts, idC1) KPRSA, (Ks)) KPBCA) to the client C;`
else
  `Send (incorrect login or password);`
end if

2) Scenario 2. Where Server B Is Nearest: In the case where B (which is partner with the server A on which the client C depends) is the nearest server (see Fig. 2). The client C sends an authentication request to B containing its hashed login and password (using MD5) all encrypted with its private key KPRC, adds its identifier to the query and then encrypts the whole by the public key of A KPBCA. The server identifier A is also added.

Algorithm 2 Service-request processing

When server B receives a service request from client C:
`Receive Service-request((`TL_{SA-CC}`, Ks, Ts, idC1) KPBSB, (idC, Iv) Ks);`
`Extract (`TL_{SA-CC}`, Ks, Ts, idC1);`
`Extract ((idC, Iv);`
$TL_{SB-CC} = TL_{SB-SA} * TL_{SA-CC}$;
if `((Iv≤Ts) ∧ (idC1 == idC) ∧ (`$TL_{SB-SA} \geq TLmin$`))` **then**
  `Send Service-response(1);` // Service granted
else
  `Send Service-response(0);` // Service refused
end if

When the server B receives the request, it transmits it to the server A, which decrypts it with its private key KPRCA. Then he decrypts it with the user's public key. Then, he compares the hash of the login and the password with those stored in his database if they correspond. If this is the case, the server A sends to the server B a positive response with a session key Ks, a session validity Ts, its identifier and its trust level TL_{SA-CC} encrypted with the server-side public key of the server B KPBSB. Plus the session key Ks encrypted with the client's public key KPBC.

When the server B receives the message, it retransmits it with the session key Ks, the TL_{SA-CC}, the user's identifier, and the session validity Ts, as well as the other message which contains the key session Ks encrypted with the client's public key KPBC (Algorithm 3).

The client wants to access a service provided by the server B, so he sends his identifier idC and the clock Iv all encrypted with the session key Ks, he has recovered, and the first message as received. Then the server B recovers the trust level TL_{SA-CC}. Then, he calculates the trust level TL_{SB-CC} using (2):

$$TL_{SB-CC} = TL_{SB-SA} * TL_{SA-CC} \quad (2)$$

where TL_{SB-SA} is the trust level of the server B towards the server A. It retrieves the session key Ks with which it decrypts the second message to thus recover the client identifier as well as the clock time Iv to compare them with the identifier and the session validity Ts in order to verify respectively the identity of the client and the possible expiration of the session. Once these tests are validated, the server B checks whether or not the trust level TL_{SB-CC} allows the user to access the service.

3) Scenario 3. Where Server B Is Nearest and Server D Provides Service: A client wants to authenticate to the server B (which is partner with the server A on which the client depends). This in the case where B is the nearest server, in order to benefit from a service of the server D (partner with B) as shown in Fig. 3. The client sends an authentication request to B containing its login and password hashed with the function hash MD5, all encrypted with its private key KPRC, adds its identifier to the request and then encrypts the whole by the public key of A KPBCA, adding the server A identifier.

When the server B receives the request, it transmits it to the server A. When A receives the request, he decrypts it with his

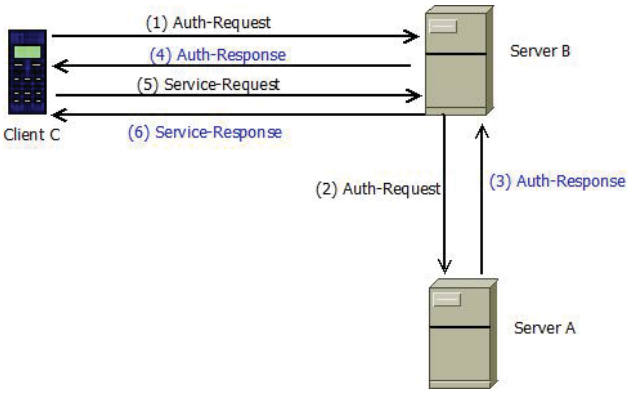

Fig. 2 Scenario 2 authentication diagram

Algorithm 3 Authentication-request processing

When server B receives an authentication request from a client C:

```
Receive Auth-request((H (login, psw)) KPRC,
idC) KPBCA;
Recover (idA);
Send (((H (login, psw)) KPRC, idC) KPBCA) to
the server A;
```

When server A receives a request from server B:

```
Receive Auth-request(H (login, psw)) KPRC,
idC) KPBCA);
Decrypt(H (login, psw)) KPRC, idC) KPBCA);
Check (idC);
Recover (KPRC);
Recover (H (login, psw));
if  ((H(login) == stored H(login)) and (H(psw)
== stored H(psw)) then
   Generate (Ks, Ts);
   Send (TL_SA-CC, Ks, Ts, idC1) KPBSB, Ks)
  to the server B;
else
   Send (incorrect login or password)
end if
```

When Server B receives the response from Server A:

```
Receive Auth-response(TL_SA-CC, Ks, Ts, idC)
KPBSB, (Ks) KPBC);
Send Auth-response (TL_SA-CC, Ks, Ts, idC)
KPBSB, (Ks) KPBC) to client C;
```

When client C receives the response from server B:

```
Receive (Ks) KPBC;
Extract (Ks);
Send Service-request((TL_SA-CC, Ks, Ts, idC)
KPBSB, (idC, Iv) Ks) to server B;
```

private key KPRCA. Then he decrypts it with the user's public key. Then, he compares the hash of the login and the password with those stored in his database if they correspond. If this is the case, the server A sends to the server B a positive response with a session key Ks, a session validity Ts and its identifier and its trust level TL_{SA-CC} encrypted with the server-side public key of the server KPBSA. Plus the session key Ks encrypted with the client's public key KPBC.

When the server B receives the message, it decrypts the message with its server-side private key KPRSB which contains the session key Ks, the TL_{SA-CC}, the user's identifier, and the session validity Ts, then calculates his trust in the user with (3):

$$TL_{SB-CC} = TL_{SB-SA} * TL_{SA-CC} \quad (3)$$

Then, it transmits the message that contains the session key Ks encrypted with the client's public key KPBC and another message that contains the session key Ks, the calculated trust level of B in C TL_{SB-CC}, the identifier of C IdC and the session duration Ts all encrypted with the server-side public key of D KPBSD to the client D (Algorithm 4).

The client wishes to access a service provided by the server D, then he sends its identifier idC and the clock Iv all encrypted with the session key Ks he has recovered, and the message that contains the session key the session duration, the identifier of C and the trust level of B in C, as received by B

Then the server D recovers the trust level TL_{SB-CC}, and calculates the trust level TL_{SD-CC} using (4).

$$TL_{SD-CC} = TL_{SB-CC} * TL_{SD-SB} \quad (4)$$

where TL_{SD-SB} is the trust level of the server D towards the server B. It retrieves the session key Ks with which it decrypts the second message to thus recover the client identifier and the clock time H to compare them with the identifier and the session validity Ts in order to verify respectively the identity of the client and the possible expiration of the session. Once these tests are validated, the server D checks whether or not the trust level TL_{SD-CC} allows the user to access the service.

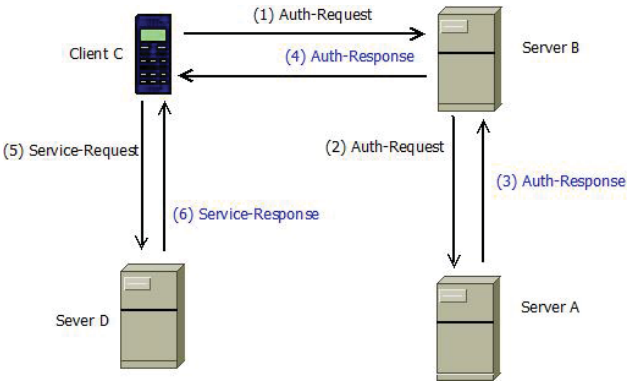

Fig. 3 Scenario 3 authentication diagram

C. Trust Management

Since a ubiquitous system usually consists of several elements of different hardware and software composition that are in constant interaction with each other and with the mobility of some, it becomes essential to manage the variations in the levels of trust between all these elements.

1) Server/Client Trust: First, we are interested in the server's trust in a client. Indeed, it is very important that a server can judge the level of trust to be granted to a client. To manage this trust, the server has in our system two metrics of trust towards a client C. The first is permanent but nevertheless

Algorithm 4 Authentication-request processing algorithm:

When server B receives an authentication request from a client C:

```
Receive Auth-request((H (login, psw)) KPRC,
idC) KPBCA, idA);
Extract (idA);
Send Auth-request (((H (login, psw)) KPRC,
idC) KPBCA) to server A;
```

When server A receives an authentication request from server B:

```
Receive Auth-request(((H (login, psw)) KPRC,
idC) KPBCA);
Decrypt Auth-request(((H (login, psw)) KPRC,
idC) KPBCA);
Check(idC);
Recover(KPBC);
Recover(H (login, psw));
if      (H (login) == stored H (login)) and (H
(psw) == stored H (psw)) then
  Generate (Ks, Ts);
  Send ((TL_SA-CC, Ks, Ts, idC1) KPBSB, (Ks)
  KPBC) to the server B;
else
  Send (incorrect login or password);
end if
```

When Server B receives the authentication response from Server A:

```
Receive Auth-response (TL_SA-CC, Ks, Ts,
idC1) KPBSB, (Ks) KPBC);
Compute TL_SB-CC = TL_SA-CC * TL_SB-SA;
Send Auth-response (TL_SB-CC, Ks, Ts, idC1)
KPBSD, (Ks) KPBC) to client C;
```

variable, and the second is temporary and not stored. Both vary according to different parameters; the first is stored at the server, it is designated by TL_{SA-CC}. The second is calculated at each authentication of the client and is stored only the time of a single session, it is designated by CTL_{SA-CC}.

Trust increases in the following cases:

1. The trust of a server A in a client C is dynamic. The context has a strong influence on trust, several elements of this context can consolidate or reduce the trust of a client C. The most obvious being the geographical position: each time the client connects to the server it records the geographic position of the client. If the client connects n times from the same geographic position, the server registers it as its familiar position. If the client connects from this position, the instant trust level CTL_{SA-CC} will be increased comparatively to TL_{SA-CC} as given in (5).

$$TL_{SA-CC} = TL_{SA-CC} * \alpha + (1 - \alpha) \tag{5}$$

α is a reel in interval $]0, 1[$ and it is taken according to criteria specific to each server.

2. After each successful authentication, the trust of the client TL_{SA-CC} increases minimally according to (5). Only in this case, α is in the range $]0.9, 1[$ and it is taken according to criteria specific to each server.

Trust decreases following equation (6):

$$TL_{SA-CC} = TL_{SA-CC} * \alpha \tag{6}$$

With α a reel that varies according to the following cases:

1) If the client connects from a position very far from where he usually connects, α is in the interval $]0.8, 1[$ and left to the discretion of the server according to its own security policies.
2) If in a short time the client moves considerably away from the last recorded point, α is in the interval $]0.5, 1[$ and decreases proportionally to the increase in distance and reduction of the time interval between the two connections.
3) The client can also be on the move. Thus, if during a single session it moves enormously, α is in the range $]0.8, 1[$ and left to the discretion of the server according to its own security policies.

Trust becomes null if a server excludes a client when the latter tries to usurp the identity of another client, or tries to break into the server, or tries to retrieve stored information that it does not have the right to access in the database. Then he will be classified as malicious.

2) Server/Server Trust: Cooperation and partnership depend on the inter-server trust, which is the main concept in our approach.

We will begin by describing the procedure to follow if an unknown server would like to become a partner with server A:

The server D wishing to become a partner with A, adds it to the list of pretenders to become partners and sends it a request (PartRequest). As soon as A receives it, it sends (Explore) requests to all its partners SP to send him their opinions on the server D. Each time a server in the set receives this request, it checks whether the server is one of his partners. If so, it sends back to the server A a message containing its trust in D. Otherwise, he sends back a message stating that he is unknown to him.

If the server A receives recommendations from its partners (the server D is partner with one of its partners), it calculates the trust level. First, each time it receives a recommendation from server i, it calculates the trust level (TL^i_{SA-SD}) using (7).

$$TL^i_{SA-SD} = TL_{SA-Si} * TL_{Si-SD} \tag{7}$$

Then calculate TL_{SA-SD} by averaging TL^iSA-SD for i from 1 to n, n being the number of recommendations received. Thus, TL_{SA-SD} will be the trust level that A will assign to the new server D. If TL_{SA-SD} is greater than the threshold (TT_{SA}), then it adds it to its list of partners and informs D by sending a response. Otherwise, he adds him to a list of pretenders where he can be promoted to a partner if his trust level increases and also informs him by sending him a response.

In case no recommendation emanates from all the partners of A (the server D is not partner with any of them) then A sends a request to its partners (DeepExplore). When the partners of server A receive it, they send a request (Explore) to their partners and treat the responses in the same way A would handle the responses. Then return the results to A.

If A receives one or more recommendations (D is a partner with one of his partners) then A calculates TL^{i}_{SA-CC} and computes the average to reduce TL_{SA-SD}. Then inform the server D that they are now partners if TL_{SA-SD} is greater than the threshold (TT_{SA}). If not, he adds it to the list of pretenders and also informs him by sending him a response.

If all the responses are negative, the server A gives a minimum trust level to D $TL_{SA-SD} = 0.1$, adds him to the list of pretenders and then informs him by sending him a response.

When the server D receives the response from A, if it is positive, then the server D adds A to its list of partners, and assigns him a trust level $TL_{SD-SA} = TT_{SD}$. Otherwise, he gives him a minimum trust level $TL_{SD-SA} = 0.1$ and adds him to the list of pretenders.

Procedure for adding a new partner is given in Algorithms 5-7.

Algorithm 5 When server A receives PartRequest (idD, idA) from server D:

SP := set of partners of A;
N := Card (SP);
Cp: = 0; // Counter
Receive PartRequest(idD, idA);
Send Explore(idD) to SP;
Wait for all responses;
if (at least one response is positive) **then**
  for i = 1 to N **do**
    if ($TL_{Si-SD} \neq Null$) **then**
      $TL^{i}_{SA-SD} = TL_{Si-SD} * TL_{SA-Si}$;
      $TL_{SA-SD} = TL_{SA-SD} + TL^{i}_{SA-SD}$;
      Cp: = Cp + 1;
    end if
  end for
  $TL_{SA-SD} := TL_{SA-SD}/Cp$;
else
  Send DeepExplore(idD) to SP;
  Wait for all responses;
  if (at least one response is positive) **then**
    for i = 1 to N **do**
      if ($TL_{Si-SD} neq Null$) **then**
        $TL^{i}_{SA-SD} := TL_{Si-SD} * TL_{SA-Si}$;
        $TL_{SA-SD} := TL_{SA-SD} + TL^{i}_{SA-SD}$;
        Cp = Cp + 1;
      end if
    end for
    $TL_{SA-SD} := TL_{SA-SD}/Cp$;
  else
    $TL_{SA-SD} := 0.1$;
  end if
end if
if ($TL_{SA-SD} > TL_{SA}$) **then**
  Add D to SP;
  Send response(1) to D;
else
  Add D to the list of pretenders;
  Send response(0) to D;
end if

Trust increases in the following cases:

1) If a server B, partner with A and with D, recommends D to A, then A calculates TL^{N}_{SA-SD} using (8):

$$TL^{N}_{SA-SD} = TL_{SA-SD} * TL_{SB-SD} \quad (8)$$

Algorithm 6 When server B partner of server A receives Explore(idD) from server A:

SP: = set of partners of B;
N: = Card (SP);
Receive Explore(idD);
i: = 1;
while i > N **do**
  if (idD == idi) **then**
    Send response(TL_{SB-SD}) to A;
  end if
  i: = i + 1;
end while
if (i> N) **then**
  Send response (0) to A;
end if

Algorithm 7 When server B partner of server A receives DeepExplore(idD) from server A:

SP: = set of partners of B;
N: = Card (SP);
Receive DeepExplore(idD);
Send Explore(idD) to SP;
Wait for all responses;
if all responses are negative **then**
  Send response (0) to A;
else
  for i =1 to N **do**
    if $TL_{Si-SD} neq Null$ **then**
      $TL^{i}_{SB-SD} := TL_{Si-SD} * TL_{SB-Si}$;
      $TL_{SB-SD} := TL_{SB-SD} + TL^{i}_{SB-SD}$;
      Cp: = Cp + 1;
    end if
  end for
  $TL_{SB-SD} := TL_{SB-SD}/Cp$;
  Send response (idD, TL_{SB-SD}) to A;
end if

Then, make the comparison with his own trust level in D TL_{SA-SD}, and his trust level in B.

If $((TL_{SA-SD} < TL_{SA-SB}) \wedge (TL_{SA-SD} < TL^{N}_{SA-SD})$
then $TL_{SA-SD} = TL^{N}_{SA-SD}$
EndIf

2) Trust can also vary according to (9):

$$TL_{SA-SD} = TL_{SA-SD} * \alpha + (1 - \alpha) \quad (9)$$

Such as α in the interval $]0.8, 1[$ and left to the discretion of the server according to its own security policies, and this in the following cases:

- If the last collaboration between the two servers A and D is satisfactory
- If the server is available on the last attempt A solicits D.
- If the number of partners in common between A and D is large.
- If during the last session the ping is short (the response time).

Trust decreases using (10), where α is as defined previously, and in the following cases:

$$TL_{SA-SD} = TL_{SA-SD} * \alpha \quad (10)$$

- If server D is not available during the last five attempts.

- If the Ping is high (the response time is large).
- If the server D recommends a user C and that the latter is revealed to be malicious (in this case the trust will decrease considerably).

V. Analysis and Comparison

Our proposal is based on the integration of trust between the entities of the system as well as the concept of partnership between servers to facilitate the authentication of the user and thus facilitate access to different services. This also allows better availability as the user authenticates to the nearest server and therefore the most available. The communications between the servers will be transparent to the user thus making the process easier. This will also allow the user to access services provided by servers other than the one on which he depends. In our approach, the notion of partnership between servers is essential because it is the basis of cooperation between servers whether to provide services or to recommend or warn a third party. It is the trust between these servers that determines the degree of their cooperation, going as far as partnership. Knowing that two partner servers are complicit to the highest point, each server has its trusted network which is made up of servers to which it has high confidence. This network is obviously dynamic; some servers can find themselves outside this circle when others can make their entry according to predefined criteria that manage this trust.

Our work has similarities to both certificate-based and identity-based trust models. Trust between two servers or between a server and a user can be done on the recommendation of another server, it is a kind of certificate; in certificate-based models, these are issued by a trusted central authority. In our model, not being practical to set up a central authority that would be solicited all the time in a ubiquitous system, we have opted for a certification of another kind that comes from a trustworthy entity (a partner server).

In addition, the servers in our model have directories on other servers they trust with of course their identities. So, as in identity-based models, these entities (servers) can be uniquely identified by their identifier, and therefore assigned a dynamic trust metric.

On the other hand, our model is based on public key management to encrypt and secure communications between servers and between clients and servers. For this, the proposed model is inspired by Kerberos [20]. Indeed, the key management is entrusted to a trusted entity that is common between the server and the client. This entity is partner with the server and the client depends on it. But, we introduce a simplification of Kerberos, because it is this (trusted) entity that delivers directly the ticket that allows access to the service, without having to use two servers as in Kerberos, where two servers (the authentication server and the server issuing tickets) share the tasks. In our model, the procedure is not centralized, indeed each server has a set of clients that depend on it, so we will have several authorities that can certify the identity of their users to a number of servers (their partners).

Our model has the following advantages:

- Promotes mobility: in a ubiquitous environment where entities are supposed to move freely, mobility is paramount. As a result, the user has the choice to authenticate with several servers and thus access different services that he needs.
- Adapted to Ubiquitous systems: the proposed approach does not dependent on a central authority that is always requested.
- Flexibility: Since the user does not have to always authenticate with the same server.
- Transparency: The communications between the servers are completely transparent to the client which considerably contributes to simplify the process.
- Distributed and dynamic Trust.
- Context-awareness: Takes into account the distance to increase or decrease the trust level.

For authentication and access to the service by the client, it is only point-to-point communications, so the number of messages exchanged at each session has a constant complexity of $O(1)$.

For trust management, the worst case scenario is adding a new partner. In this case, the propagation of the request to the partners (and possibly the partner's partners) of the requested server will require a number of messages of the order of $O(n)$, it is the same complexity as for the response, whether it is positive or negative. Here, "n" represents the number of nodes of the entire concerned network.

Nevertheless, our model remains simple and is still at a primitive stage of its design. In addition, the criteria and conditions for managing trust can be enriched. Our proposal does not, yet, include mechanisms for detection of a malicious server and its exclusion.

VI. Conclusion and Perspectives

The main purpose of computers being to facilitate the tasks of everyday life, the ubiquitous systems come to contribute a little more. Indeed, the principle of ubiquitous computing is to be present everywhere and to continually adapt to the user's context and needs. Securing these systems is evident, even though tremendous progress has been made in this direction, there is still much to be done because it is a vast field and the challenges remain multiple.

Our proposal develops an authentication model adapted to context-aware systems based on trust. We defined this notion of trust and that of partnership as well as the rules of their management making them dynamic and distributed. Each server within our system maintains its own levels of trust relative to and dependent on other servers. Our proposal is based on the principle of a trusted third party but with a distributed and more flexible approach.

As a perspective, future work will focus on:

- Include other situations for trust management such as excluding a server from a trusted network.
- Improve the formula for the assessment of trust because it is simple and does not take into account certain parameters.
- Allow servers that detect a threat to be able to warn their partners and manage the reaction of the latter.
- Realize an algorithm that would launch a vote to make a collective decision to exclude a server.

- Adapt our system for better directory management when this system is applied on a large scale.

REFERENCES

[1] M. Weiser, "Ubiquitous Computing ", 1996, http://www.ubiq.com/ hypertext/weiser/ UbiHome.html, (06/12/2017).
[2] A. Mansour, M. Sadik, E. Sabir, and M. Azmiy, "A Context-Aware Multimodal Biometric Authentication for Cloud-Empowered Systems," in Proc. of International Conference on Wireless Networks and Mobile Communications (WINCOM), 26-29 Oct., Fez, Morocco, 2016.
[3] A. Chaturvedi, A. K. Das, D. Mishra, and S. Mukhopadhyay, "Design of a secure smart card-based multi-server authentication scheme," Journal of Information Security and Applications, Volume 30, Issue C, pp. 64-80, October 2016.
[4] D. Hintze, S. Scholz, E. Koch, and R. Mayrhofer, "Location-based Risk Assessment for Mobile Authentication," UBICOMP/ISWC-16 ADJUNCT, pp. 85-88, Heidelberg, Germany, September 12-16, 2016.
[5] U. S. Premarathnea, I. Khalil, and M. Atiquzzaman, "Location-dependent disclosure risk based decision support framework for persistent authentication in pervasive computing applications," Computer Networks, Vol. 88, pp.161-177, 2015.
[6] B. Shivhare, G. Sharma, and S. P. S. Kushwah, "A Study On Geo-Location Authentication Techniques," 2014 Sixth International Conference on Computational Intelligence and Communication Networks, CICN 2014, Bhopal, India, pp. 744-748, 14-16 November 2014.
[7] M. A. Bouazzouni, E. Conchon, and F. Peyrard, "Trusted mobile computing: An overview of existing solutions," Future Generation Computer Systems, Volume 80, pp.596-612, March 2018.
[8] H. Xiao, J. Malcolm, B. Christianson, and Y. Zhang, "Trustworthiness and Authentication in Ubiquitous Computing," in Proceedings of MobiWac-12, Paphos, Cyprus, pp.135-138, October 21-22, 2012.
[9] G. Sarojini, A. Vijayakumar, and K. Selvamani, "Trusted and Reputed Services using Enhanced Mutual Trusted and Reputed Access Control Algorithm in Cloud," 2nd International Conference on Intelligent Computing, Communication and Convergence (ICCC-2016), Bhubaneswar, Odisha, India, Procedia Computer Science, Vol.92, pp.506-512, 2016.
[10] K. Selvamani and P. K. Arya, "Credential Based Authentication Approach for Dynamic Group in Cloud Environment," International Conference on Intelligent Computing, Communication and Convergence (ICCC-2014), Bhubaneswar, Odisha, India,Procedia Computer Science, Vol.48, pp.166-172, 2015.
[11] R. Shaikh and M. Sasikumar, "Trust Model for Measuring Security Strength of Cloud Computing Service," International Conference on Advanced Computing Technologies and Applications (ICACTA- 2015), Procedia Computer Science, Vol. 45, pp. 380-389, 2015.
[12] S. Arimura, M. Fujita, S. Kobayashi, J. Kani, M. Nishigaki, and A. Shiba, "i/k-Contact: a context-aware user authentication using physical social trust," Twelfth Annual Conference on Privacy, Security and Trust (PST), Toronto, Canada, pp. 407-413, 23-24 Jul 2014.
[13] Q. G. K. Safi, S. Luo, C. Wei, L. Pan, and G. Yan, "Cloud-based security and privacy-aware information dissemination over ubiquitous VANETs," Computer Standards and Interfaces, Vol. 56, pp. 107-115, February 2018.
[14] S. Jain and A. Ranjan, "A Review Study on Vehicular Ad-Hoc Networks Trust and Authentication Mechanisms," International Journal of Technical Research (IJTR) Vol. 5, Issue 1, pp. 101-106, 2016.
[15] V. Radhaa and D. Hitha Reddya, "A Survey on Single Sign-On Techniques," Procedia Technology, Vol.4, pp. 134-139, 2012.
[16] R. Saadi, The Chameleon: Un Système de Sécurit17 pour Utilisateurs Nomades en Environnements Pervasifs et Collaboratifs. PhD thesis, Institut National des Sciences Appliquées (INSA) de Lyon - France, 2009.
[17] L. Rasmussen, A. Rasmussen, and S. Janson, "Reactive Security and Social Control," 19th National Information Systems Security Conference, Baltimore - USA, 1996.
[18] Y. Wang and J. Vassileva, "Trust and Reputation Model in Peer-to-Peer Networks," in proceedings of Third International Conference on Peer-to-Peer Computing, (P2P 2003). Linkoping, Sweden 1-3 Sept. 2003.
[19] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD," Shanghai Jiaotong University, Shanghai - Chine, 2004.
[20] J. Garman, Kerberos, The Definitive Guide. Edition O'Reilly, 2010.