International Journal of Soft Computing 2 (1): 157-160, 2007

Detail Analysis on the Authentication Models and Developing a Framework for Distributed Trust Model in Internet Scale Ubiquitous Computing

A.K.M. Najmul Islam and K.M. Najmul Islam
Insinöörinkatu 60 C 181, 33720 Tampere, Finland

Abstract: Ubiquitous computing aims at defining environments where human beings can interact in an intuitive way with surrounding objects. Those objects, which can be personal digital assistants, electronic rings, doors or even clothes, offer embedded chips with computation power and communication facilities and are generally called artifacts. Apparently confidentiality of communication seems to be the most important issue as the prime concern for any system relying on wireless transmission. But the real problem is actually security. Authentication is also one of the most interesting security problems in ubiquitous computing. Many researchers have proposed different suitable authentication protocols for ubiquitous environment. But if we consider the ubiquitous computing environment, in some cases we find that the authentication procedure will not work. In this study we must have to find out an alternate solution. In this study I have tried to find out some situations where the authentication protocols will not work and presented an alternate solution in these situations.

Key words: Ubiquitous computing, *ad hoc* networking, authentication, access control list, encryption, distributed trust

INTRODUCTION

With ubiquitous Computing (Ubicomp) we refer to a scenario in which computing is omnipresent[1] and particularly in which devices that do not look like computers are endowed with computing capabilities. A computer on every desk does not qualify as ubiquitous computing; having data processing power inside light switches, door locks, fridges and shoes, instead, does. We envisage a situation in which all those devices are not only capable of computing but also of communicating. We do not however expect a fixed networking infrastructure to be in place. The devices will have to communicate as peers and form a local network as needed when they recognize each other's presence. This is what we mean by *ad hoc* networking: a peer-to-peer network (no centralized server) set up temporarily to meet some immediate need. In this paper I am proposing a distributed trust model, which can provide good security.

EXISTING AUTHENTICATION MODELS

TESLA: TESLA[2] stands for Timed Efficient Stream Loss-Tolerant Authentication. It is a multicast authentication protocol. TESLA takes into account the computational overheads associated with asymmetric key cryptography and hence employs the less expensive MAC (Message Authentication Code). The requirement of this approach is that the communicating peers should be loosely time synchronized. TESLA makes use of self authenticating one way chains. One way chains are a widely used cryptographic primitive.

TESLA is applicable for multicast data streaming applications where one way authentication is required. Consider the case in which a manager has to send the meeting agenda to all the personnel attending the meeting. The receivers have to make sure that the information is sent from the manager and that it has not been sent by any unknown person who may just be aiming to misguide the employees. The sender initially commits to a secret key K with the receiver using the traditional signature method. The sender in the next packet uses this key K as the MAC key and appends the message with its MAC and sends it to the receiver.

The receiver on reception buffers the packet until the sender discloses that key. On disclosure of the key, the receiver checks if the key is a correct one using self authentication and the previously released key. It then checks the correctness of the MAC and accepts the packet if the MAC is valid. The TESLA protocol is simple and is a good candidate for our environment for the following reasons:

Corresponding Author: A.K.M. Najmul Islam, Insinöörinkatu 60 C 181, 33720 Tampere, Finland

- Low computational overhead.
- Low communication overhead.
- Scales to a large number of receivers.
- Additionally provides some data integrity by the mechanism of validating MACs.

The disadvantages of this approach are:

- Loose time synchronization.
- Need to buffer messages till the key is disclosed.
- Initial authentication i.e., for the first time principals communicate, they have to exchange some secret information for forming the same chain keys and this requires digital signatures.
- Requires timely delivery of packets else they are subjected to rejection meaning that network congestion is not accounted for.
- Does not provide non repudiation.

µTESLA and SNEP: Due to the limitation of TESLA, SPINS[3] (Security Protocols for Sensor Networks) came up with an extension of TESLA called µTESLA and another protocol called SNEP (Secure Network Encryption Protocol). SNEP provides data confidentiality, authentication and integrity and data freshness. Initially a shared key is established between the communicating principals using some bootstrap approach[4]. The sender and receiver also maintain a counter C whose state is maintained at both the sides. The master key is then used to derive two more keys; the encrypting key Kencr and the MAC key Kmac. If A wants to communicate message M to B, the message sent is,

$$A \rightarrow B : \{M\}_{(kencr,C)}, MAC[Kmac, C \text{ i } \{M\}_{(kencr,C)}]$$

On reception, if the MAC verifies correctly, the authentication of sender is achieved.
SNEP provides the following advantages:

- Low communication overhead, adds only 8 bytes per message.
- Achieves semantic security.
- The attacker cannot replay messages due to presence of counter.

However it requires the initial bootstrapping to be done whereby secret key is exchanged, the method of which is not definitive. µTESLA extends the efficiency of TESLA for broadcast authentication by making use of just symmetric mechanisms.

The resurrecting duckling model: The Resurrecting Duckling model[5] presents a different approach to the environment. Every device is known to be a duckling and has a mother duck (its owner). The mother ducks, soon after the birth of the duckling, *imprints* it. Imprinting is the method of acquiring ownership of the device. Without going much into the interesting analogies made by this model, I will highlight the approach to authentication here. The model assumes the complete absence of any online server availability and hence precludes the use of approaches like Kerberos[6]. In their approach, the duckling and its mother duck share a secret key much like we saw in SNEP. However this secret key establishment is done via a physical contact which they (F Stajano and R Anderson) portray as both cheap and effective. Having obtained the secret key, symmetric key protocols can be employed[7,8].

Islam's model: In Islam[9] model every device will be imprinted by its owner by a secret key. This secret key owner id. Each device would also have a unique identification (device id). The device id will be an IPv6 address rather than IPv4. Islam[9] considers the devices of the same owner under a unique group. Each device will have an access control list which will hold the device ids under the same group.

Every time a new device bought by the owner, he needs to manually store all the device ids of his group into the access control list. So Islam[5] keeps provision for sharing the access control list by a new device.

Authentication prior to any communication is performed in the following way:

- Any sender wishing to communicate will send a communication request message to the receiver. This request may be captured by anyone in the transit.
- Whoever the receiver may be (as the receiver is not authenticated) it will send two prime numbers, p and q, to the sender for using RSA encryption.
- Sender will generate a random number,R1 and encrypt it using its owner id to produce the message, M1.

$$M1 = Encrypt_{owner\ id}(R1)$$

The encryption can be done by bit level encryption technique (Exclusive OR).

- M1 will then be RSA encrypted and sent to the receiver.

$$M1_{RSA} = Encrypt^{RSA}(M1)$$

- On reception of $M1_{RSA}$ receiver will decrypt it by RSA technique to generate M1.

$$\text{Decrypt}^{RSA}(M1_{RSA}) = M1$$

- Receiver will then decrypt M1 using its owner id to get a number, R2.

$$\text{Decrypt}_{\text{owner id}}(M1) = R2$$

- R2 will be sent to the sender using RSA encrypted message, M2.

$$M2 = \text{Encrypt}^{RSA}(R2)$$

- Sender will decrypt M2 using its owner id.

$$\text{Decrypt}^{RSA}(M2) = R2$$

If R2 = R1 then sender becomes sure that the receiver belongs to the same group as its own. This way the receiver is authenticated. Steps 3 through 8 are followed for authenticating the sender in the same way.
The model provides the following advantages:

- This seems that our proposed model is able to provide a much secured authentication under any circumstances; an imposter can never get authenticated.
- The model provides on the fly authentication because here in this case there is no traditional password giving mechanism.
- It supports both way authentications: sender gets authenticated by the receiver and vice versa.
- It uses RSA encryption and thus prevents man in the middle attack.
- It uses no central authentication server and this way avoids two problems associated with central approach, namely- single point of failure and performance bottleneck.

It has the following disadvantages

- The proposed model is very much computation oriented as it uses RSA encryption.
- It does not support multiple ownership of a device.

Some common problems with the authentication models: When considering a *Smart Office* scenario, where intelligent services are accessible to mobile users via hand-held devices connected over short range wireless links, I encountered several problems with security for *such systems*. Firstly, it is not possible to have a central authority for a single building, or even a group of rooms. So I have to use a distributed model, where the *service managers*, each of which are responsible for a subset of services, are arranged in a hierarch[11,8]. But the above-mentioned authentication models are not sufficient to authenticate users because most of the times users are foreign to the system, i.e., they are not known. Consider a *Smartroom* in an office, equipped with an MP3 player, fax machine, lights, fans, a coffee maker and a printer. If a person with his laptop is known to the system, he is allowed to access the services of the *smartroom* after being authenticated by Islam's proposed protocol[9]. But if a user, Fahim, walks, how does the room decide which services Fahim has the right to access. Just authenticating Fahim's certificate gives no information on access control because Fahim is an unknown user. Unless it is known in advance which users are going to access the room and their access rights are also known, Islam's proposed authentication and access control[9] is not going to work.

PROPOSED DISTRIBUTED TRUST MODEL

Trust architecture: A *security policy* is a set of rules for authorization, access control and trust in a certain domain[12]. All services/users of the domain must enforce its policy and can impose a *local policy* as well. A service being accessed by a foreign user should verify that the user conforms to both its policies. The policy in each domain is enforced by special agents called *security agents*. These agents are part of the Service Manager. Users/agents are identified by authentication certificates. Delegations can be made by authorized agents in the form of signed assertions. Security agents are able to reason about these signed assertions and the security policies to provide access control to the services in their domain.

In this system consider 'delegation' as a permission itself. Only an agent with the right to delegate a certain action can actually delegate that action and the ability to delegate, itself can be delegated. Delegations can be constrained in the policy, by specifying whether an agent can delegate a certain right and to whom it can delegate. Rights or privileges can be given to trusted agents, who are responsible for the actions of the agents to whom they subsequently delegate the privileges. So the agents will only delegate to agents that they trust. This forms a delegation chain. If any agent along this chain fails to meet the requirements associated with a delegated right, the chain is broken and all agents following the failure are not permitted to perform the action associated with the right.

Agents can make requests for a certain service to a security agent controlling the service and while doing so they attach all their credentials, i.e., ID certificate, authorization certificates etc., to the request. The *security agents* generate authorization certificates, that can be used as 'tickets' to access a certain resource. An agent can also request another agent to delegate to it the right to access a certain service. The latter agent, if satisfied with the requester's credentials may decide to send back a signed statement containing the delegation. The security agent is responsible for honoring the delegation, based on the delegator's and delegatee's credentials and the policies. The security policy could also contain information about roles of some agents and the abilities associated with certain roles.

Assume Fahim does not work in the office, but in one of its partner firms. How will the system decide whether to allow him to use certain services? I have thought *Distributed Trust* as the solution.

Fahim is an employee of one of the office's partners, but the service manager is unable to understand his role in the organization, so he is denied access to the services. Fahim approaches one of the managers, Emon and asks for permission to use the services in the *SmartRoom*. According to the policy, Emon has the right to delegate those rights to anyone he trusts. Emon delegates to Fahim, the right to use the lights, the coffee maker and the printer but not the fax machine, for a short period of time. Emon's laptop sends a short lived signed delegation to Fahim's hand-held device. When Fahim enters the room, the client on his hand-held device sends his identity certificate and the delegation to the service manager. As Emon is trusted and has the ability to provide delegation, the delegation conforms to the policy and Fahim now has access to the lights, the coffee maker and the printer in the room. Once the delegation expires, Fahim is denied access to any service in the room and must ask Emon for another delegation. In this way, a foreign user, Fahim, is allowed access to certain services without creating a new identity for him in the system or assigning a temporary role to him or insecurely opening up the system in anyway.

Now if we look at the Internet Scale Ubiquitous Computing, we can see that the concept of *distributed trust* can play an important role in the overall security of the system.The authentication models can be adopted only with small ubiquitous environment. But in the real situation the environment is very large and in this case only distributed trust can provide the desired solution.

CONCLUSION

I have proposed a trust architecture in this study. This model is very much suitable for Internet Scale Ubiquitous Computing. The trust architecture is very much difficult to implement in the case of such large systems. So the trust architecture should be reviewed so that the performance is increased. My next target is to improve the trust architecture.

REFERENCES

1. Stajano, F., 2002. Security for Ubiquitous Computing, Uk.
2. Perrig, A., R. Canetti, J.D. Tygar and Dawn Song, 2000. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. Security and Privacy, 2000. S and P; 2000. Proceedings. 2000 IEEE Symposium, pp: 56-73.
3. Adrian Perrig, Robert Szewczyk, J.D. Tygar, Victor Wen and David E. Culler, 2002. SPINS: Security protocols for sensor networks. Wireless Networks 8: 521-534.
4. Helton, R. and J. Helton, 2002. Mastering Java Security: Cryptography Algorithms and Architecture, USA.
5. Stajano, F. and R. Anderson, 2002. The Resurrecting Duckling: Security issues for ubiquitous computing. Computer, IEEE J., pp(s): supl22-supl26.
6. Neuman, B.C. and Ts'o T, Kerberos, 1984. An authentication service for computer networks. IEEE Communications Magazine, pp: 33-38.
7. Marc Langheinrich, 2002. A Privacy Awareness System for Ubiquitous Computing Environments, UbiComp, pp: 237-245.
8. Colin English, Paddy Nixon, Sotirios Terzis, Andrew McGettrick and Helen Lowe, 2002. Dynamic Trust Models for Ubicomp Environments, in Ubicomp2002 Security Workshop, Göteborg.
9. Islam, 2005. An efficient model for on the fly authentication in ubiquitous computing system. Asian J. Inform. Technology.
10. Al-Muhtadi, J., A. Ranganathan, R. Campbell and M.D. Mickunas, 2002. A flexible, privacypreserving authentication framework for ubiquitous computing environments, Distributed Computing Syste ms Workshops, 2002. Proceedings, 22nd International Conference, pp(s): 771-776.
11. Wood, A.D. and J.A. Stankovic, 2002. Denial of service in sensor networks, Computer, pp(s): 54-62.
12. Victoria Bellotti and Abigail Sellen, 1983. Design for Privacy in Ubiquitous Computing Environments, ECSCW, pp: 75.