The Application of Fuzzy Logic Controller to Compute a Trust Level for Mobile Agents in a Smart Home

Sandy Nasution, Nanna Suryana, Shahrin Shahib, Nur Azman Abu, Pieter Hartel
Universiti Teknikal Malaysia Melaka
Melaka /Malaysia
{sandy_ra,nsuryana,shahrinsahib,nura}@utem.edu.my, pieter.hartel@utwente.nl

Abstract

Agents that travel through many hosts may cause a threat on the security of the visited hosts. Assets, system resources, and the reputation of the host are few possible targets for such an attack. The possibility for multi-hop agents to be malicious is higher compared to the one-hop or two-hop boomerang agents. The travel history is one of the factors that may allow a server to evaluate the trustworthiness of an agent. This paper proposes a technique to define levels of trust for multi-hop agents that are roaming in a smart home environment. These levels of trust are used later to determine actions taken by a host at the arrival of an agent. This technique uses fuzzy logic as a method to calculate levels of trust and to define protective actions in regard to those levels.

Keywords
Smart Home, mobile agents, security, shortest path, fuzzy logic controller.

1. INTRODUCTION

Home Networks are becoming a reality. In the near future, people will have many devices that communicate with each other in their home [1]. At first, there was only a single personal computer (PC) in the home without a connection to the outside world. Now, most homes have many computers with an Internet connection to the outside world. In the future, we can expect that a large population of networked devices that will support us in the home. These devices run mobile agents which provide the home occupants with services.

The agents require a number of functionalities that are specific to a smart home scenario. To build those functionalities requires the integration of technologies from databases, robotics, machine learning, mobile computing, and multimedia computing. The desired smart home functionalities must be organised into an architecture that seamlessly connects these mobile agents while allowing improvement in any underlying technologies. The functionality of agent is provided by four cooperating layers. The decision layer selects actions for agents to be executed based on information supplied by the other layers through the information layer. The information layer gathers, stores, and generates knowledge useful for decision making. The communication layer facilitates the communication of information request between agents. Finally, the physical layer contains the hardware including individual devices and network hardware [2].

Mobile agents offer a possibility to encapsulate information of a person and the person's preferences and to request location based services of the system on behalf of the user. Security and privacy are a major concern for such an agent system [11]. Since, smart home appliances are easily plugged into smart home networks, the question of how to perform security management arises, especially regarding authentication, identification, secure messaging, certification, trusted third party involvement, and resource control for such appliances arises [3]. The problem is compounded by the fact that there is no system administrator in the smart home.

Different kinds of attacks are possible on a mobile agent system. Since a mobile agent system consists of mobile agents and the mobile agent platform (host), an attack on the host by malicious mobile agents, and an attack on a mobile agent by a malicious host are the two main threats to a mobile agent system. A mobile agent that has been affected and turned into a malicious agent might take resources, confidential data or utilize the host as the basis of new attack. On the other hand, the hosts where the agents visit can also be malicious and can damage or spy on the visiting agents.

Ordile [10] explains that the central security concern for systems where agents roam is how to establish trust; she argues that the travel history of the agent is the key to establish trust. Security for agents with constrained travel itineraries is simpler to manage than security for general multi-hop agents. This is because the one-hop and two-hop boomerang agents engage with a limited number of hosts during their travels and the hosts are mostly known in advance. It is unreasonable to expect widely travelled agents to know their routes and destination in advance, because they may be searching for new information, learning as they travel. Therefore, in the same paper Ordille proposes to use a path history to tackle such problems. The general idea of a path history is to remember the hosts visited by the agent. When an agent arrives at a host, the authenticated list of previously visited hosts can be inspected. In addition, the host will ask for further information about the agent before deciding on the level of trust. The remaining problem is then how to map a path history into a level of trust.

Protecting a mobile agent platform from malicious agents is an active area of research. *Protection mechanisms* such as sandboxing, code signing, and proof carrying code; safe languages such as Java, Safe_TCL, and Typescript, as well as safe operating systems such as Singularity provide ways of protecting agent platforms against malicious agents. However, it is difficult to deploy the right protection techniques since each must be properly implemented and properly used [10]. In this paper, we focus on the latter issue: how to decide when to use a particular protection mechanism.

There are many decision factors that may cause a mobile agent to become malicious. Ordille suggests that the path travelled by a multi-hop agent can be taken as one of the factors [10]. Grimley suggests that time is also one of the *decision factors* since the more time an agent spends executing, the more chance it has of becoming corrupted. The decision mechanism should take its decision on the basis of all the relevant decision factors, which we might include, time and path, but also others, such as the past behaviour of the agent.

The problem addressed in this paper is how to combine the *decision factors* gleaned from various sources in a “level of trust” that the platform may have in the agent, and then for the platform to use that level of trust in such a way that the appropriate *protection mechanism* can be activated.

Our contribution consists of a proposal to use a Fuzzy logic based decision mechanism, that on the one hand provides the necessary flexibility to accommodate all the relevant decision factors, and on the other hand naturally deals with the inherent fuzziness of the concept of “level of trust”. Fuzzy logic has proved to be successful in many practical applications, but to the best of our knowledge fuzzy logic has not been applied to the decision making process for the security of a mobile agent platform. We assume the reader to have a basic understanding of Fuzzy logic, as for instance provided in a standard text book [14].

The paper is organized as follows: The literature on the existing protection mechanism for mobile agents is reviewed in section two. The decision factors proposed in the literature are reviewed in section three. The proposed security model using the Fuzzy Logic controller is described in section four. The conclusion and the future work of this research are described in the last section.

2. PROTECTION MECHANISMS

There are several protection mechanisms that can prevent attacks on mobile agents from malicious hosts and vice versa. This paper is focus on protecting the host from malicious mobile agents. The most important mechanisms are briefly summarized below.

2.1 Sandboxing

Sandboxing is a software technique used to protect a mobile agent platform from malicious mobile agents. In an execution environment (platform), local code is executed with full permission where it has access to all system resources. On the other hand, remote code, such as mobile agents and downloadable applets, are executed in a restricted area called a "sandbox". The restriction affects certain code operations such as interacting with the local file system, opening a network connection, accessing system resources on the local system, and invoking a program on the local system. This ensures that a malicious mobile agent cannot cause any harm to the execution environment that is

running it [5]. The drawback of this technique is that a failure in any of interrelated security parts may lead to a security violation, and it also increases the execution time of legitimate remote code.

2.2 Code Signing

The "code signing" technique ensures the integrity of the code downloaded from the Internet. It enables the platform to verify that the code has not been modified since it was signed by its creator. The advantage of this technique is the ability to verify that the code sent between agents has not been modified. The drawback of code signing is the inability to reveal what the code can do or guarantee that the code is in fact safe to run [5].

2.3 Proof - Carrying Code

Lee and Necula [6] introduced the Proof-Carrying Code (PCC) technique in which the code producer is required to provide a formal proof that the code complies with the security policy of the code consumer. The code producer sends the code together with the formal security proof in the form of a machine-checkable proof, to the code consumer. When received, the code consumer checks and verifies the proof of the incoming code by using a simple and fast proof checker. The advantage of this technique is that it guarantees that the incoming code complies with the security policy of the code consumer, provided that there is no flaw in the verification condition generator, and the proof-checker. The drawback of this technique is the potential size of the proof, and the time consumed in the proof-validation process.

2.4 State Appraisal

This technique calculates the maximum set of safe permissions that the agent may request from the host platform, depending on the current state of the agent. The author of the agent needs to anticipate possible harmful modifications to the agent's state and to counteract them within the appraisal function. The advantage of this technique is that it provides a flexible way for an agent to request permissions depending on its current state and on the task that it needs to do on the particular platform [7, 5]. The disadvantage is the difficulty to formulate appropriate security properties for the mobile agent and to obtain a state appraisal function that ensures those properties [5].

3. DECISION FACTORS

When the agent platform decides to what extent it can trust the incoming agent it must base its decision on appropriate factors. Here we discus a number of proposals for such factors, and suggest one of our own (ordered shortest path).

3.1 Application information

The primary purpose of agents is to gather information on their travels, so the information collected is one of the most important inputs for the decisions making process. The nature of this information depends on the application scenarios, so it should be as easy as possible to adapt the behaviour of the platforms to the nature of the agent information.

3.2 Path History

When an agent travels a multi-hop itinerary, it visits many platforms that are not all trusted to the same extent. The newly visited platform may benefit from the answers to the following questions: where has the agent been? How likely is it that the agent has been converted to a malicious agent during its trip? To enable the platform to answer these questions, a mobile agent should maintain an authenticated record of previously visited platform during its travel history. Using this history, the platform can make the decision whether to run the agent and what level of trust, services, resources and privileges should be granted to the agent [9,10]. Ordille suggests two techniques, in the first one, each host, A_i, adds itself to the path history of an agent and signs the complete path. The destination host, A_{i+1}, verifies the signature and determines whether it can trust the agent after it has visited all the horsts in the path

history. In the second technique, host A_i that forwards the agent to another host, $A_{i+1,}$ signs an annotation for the next hop $A_i \rightarrow A_{i+1}$. A_i supplies the list of signed and A_{i+1} authenticates the path of the agent by authenticating each signed hop annotation. These two techniques make it possible to calculate a trust level for agents that are roaming a network**.**

3.3 Time

Grimley and Monroe [8] use the factor of time to help identify a malevolent host. If the amount of time needed to execute a mobile agent on a host is limited, then the chance that it would become malicious is assumed to be minimized. Once the maximum amount of time needed by a mobile agent to execute safely on an entrusted host is exceeded, the agent must shut down or move to the next host specified on its itinerary. Therefore, the execution time is a useful decision factor.

3.4 Path Patterns

There are two types of mobile agent; a one-hop-agent and a multi-hop agent. A One-hop agent is an agent that only migrates from one host to another and back, or it also can stay at the destination host. On the other hand, a multi-hop agent is a mobile agent that travels through many hosts [10]. For a multi-hop agent, the list of hosts visited is the path history. Cao and Lu [12] represent the path history as a sequence of host identifiers like "$h_1 h_2 \ldots h_n$". A path pattern is a regular expression that uses host identifiers as its alphabet to match a set of paths with some property. For example, "$*h_a h_b *$" matches the paths of the agents which have travelled to h_a and then directly to h_b.

3.5 Ordered Shortest Paths

We believe that it is also useful to be able to reason about whether an agent has taken the shortest path, the second shortest etc. The shortest path represents the lowest risk for the agent; the second shortest path represents the next best path etc. The shortest path through a network is readily provided by the routing layer, for example by the open shortest path first (OSPF) protocol [13]. It is therefore reasonable to assume that each host knows the shortest path to other hosts.

3.6 Agent Behaviour

We believe that the behaviour of an agent is also a reasonable factor to be considered. An agent that is performing its task according to its normal routines is given a good mark, but an agent that is acting differently is given a lower mark. An agent with good behaviour is an agent that works in a disciplined manner, learning while it travels. An agent behaving in a normal fashion always uses a correct path and a reasonable amount of processing time to execute, while an agent is deemed to be bad if it behaves anomalously, for example when it makes a detour on the Internet for no good reason.

4. DECISION MAKING USING A FUZZY LOGIC CONTROLLER

A Fuzzy Logic Controller (FLC) focuses on what the system should do (prescriptive scheme) rather than trying to understand how it works (descriptive scheme). This almost invariably leads to quicker and cheaper solutions to complex control problems. A FLC also offers several unique features that make it a good choice for control problems and decision making. An FLC, which processes user-defined rules governing the target system, can be modified easily to improve or alter the performance of a system. An FLC can work with fuzzy parameters, which will keep the complexity low. Because of the rule based nature of the FLC, any reasonable number of inputs can be processed and numerous outputs generated [14]. A FLC consists of a rule base and a set of membership functions. The rules are usually in the familiar 'IF-THEN' format.

We show in a number of examples how an FLC can be used to make decisions on the trust level of an agent using relevant decision factors as input. The output of the FLC is a prediction on the level of trust attributed to the agent. We have classified the level trust in three categories: High, Medium, and Low. This classification is then used by the host to determine the requirements put on the visiting agent, as well as the services, and resources granted to the visiting agent. If the trust level of the agent is High, the agent is granted access to all resources of the host, the computing resources, memory usage, and only simple authentication is required. If the agent is deemed malicious as

indicated by a Low trust level, the authentication process for this agent is sophisticated, access to resources, and memory usage is limited, and an appropriate protection technique is applied to this agent, for example sandboxing. A sample classification is shown in Table 1.

Agent trust level	Authentication Process	Data access	Memory usage	Security mechanism
High	Simple	Granted to all resources	It's allowed to use memory as much as it requires.	None needed
Medium	Normal authentication process	Granted a sufficient data access	Standard memory usage	None needed
Low	Sophisticated authentication process	Cannot retrieve crucial data	Granted limited memory	Agent execution is using the sandbox

Table 1. Agent trust level classification table

We now give a scenario and three examples to illustrate how a FLC can help to decide on the appropriate trust level, and thus to protect the integrity of the smart home.

4.1 Scenario

Consider a smart home scenario where the home owner, Dieva and her family, are out on vacation. Dieva's friend Sita is enrolled in the biometric front door lock, so that she can look into the house from time to time while Dieva is away. This is what happens the first time that Sita goes to Dieva's home:

1. Sita arrives at the front which is noted by the "guest" agent, who will serve Sita while she is in the house. The initial trust level of the guest agent is low, but this is enough to start the biometric authentication. Sita's finger print is recognised, and the door opens so that she can enter the house.
2. Sita is let in and she is thirsty, so she wants to go to the kitchen searching for water. The kitchen door biometric lock does not recognize Sita since it is their first encounter, so Sita cannot enter the kitchen. The fact that Sita has been able to open the front has given the guest agent an initial level of trust (Low), which unfortunately for Sita is too low to override the biometric door lock.
3. Sita watches TV for a while. The process is the same as for the kitchen lock, i.e. the guest agent travels through the home-control system to the TV, but this time the low trust level is sufficient to turn the TV on.
4. While watching TV, the smart phone rings, and for the same reason as with the TV, Sita is allowed to answer the phone: it is Dieva and they talk briefly. Because of the voice recognition in the Smart phone, the latter infers that Sita is talking to Dieva, the home owner. Therefore, the level of trust of the guest agent will be raised to Medium.
5. After talking to Dieva, Sita is getting really thirsty and tries to get some refreshment from the kitchen again. Thanks to the interaction with the smart phone, the guest agent is now endowed with a medium level of trust and can now override the biometric lock on the kitchen door.

The guest agent that travels through the home on behalf of Sita gathers crucial information for the decision making process. The FLC bases its decision about the trust level of the guest agent on the information gathered. Here we show an example of how processing time, path length, and an abstraction of past agent behaviour are used to decide on the appropriate trust level. As stated in the previous section the processing time for a mobile agent in a host should be limited [8]. Therefore, for this example, we assume 60 seconds as the maximum processing time and 1 second as the minimum processing time. Since the home is small, we assume that a short path is 1 hop and a long path is 10 hops. Therefore, the linguistic variables are: Processing Time = { Fast, Slow }, Path Length = { Long, Short }, Agent Behaviour = {Bad, Normal, Good} and the Trust Level = { High, Medium, Low }. To illustrate the flexibility of our FLC, we analyze three decision cases from the given scenario.

4.2 The FLC based decision engine

In this section we present the membership functions and the rule base of the FLC that are used at the three decision cases, as well as the results of the decision making. The flexibility of an FLC makes it easy to add more rules and linguistic variables to cater for other scenarios. The three decision cases are:

Case 1: Decision at the Front door. We show by evaluating the rules for the guest agent with an empty path that the trust level is Low. This is enough to start the biometric authentication.

Case 2: Second decision at the Kitchen door. We show by evaluating the rules for the guest agent with an extended path that the trust level is Medium. This is enough to override the biometric authentication and gain entrance to the kitchen.

Case 3: Third decision at the Kitchen door. This time we assume for illustration purposes that the agent behaves suspiciously by going out to the Internet before migrating to the smart phone. The rules for the guest agent with an even more extended path show that the trust level is now Low, which is not enough to override the biometric authentication at the kitchen door.

The parameters chosen for the membership functions and the fuzzy output for each of the three decision cases is shown in Table 2.

Input								
Parameters		μ	σ	w	K	Case 1	Case 2	Case 3
Time	Fast	0	25.48	1	-0.5	-	-	-
	Slow	60				40	50	60
Path Length	Short	0	4.247	1	-0.5	2	5	-
	Long	10				-	-	6
Behaviour	Good	100	19.11	1	-0.5	-	80	-
	Normal	55				30	-	-
	Bad	10				-	-	10
Output								
Trust Level	High	100	-	-	-	-	-	-
	Medium	55	-	-	-	-	43.0	-
	Low	10	-	-	-	26.0	-	2.3

Table 2. Parameters of the membership functions $FS_i(X) = e^{-k_i \cdot w_i \cdot (\frac{X-\mu_i}{\sigma_i})^2}$, where *i* ranges over the set{Fast, Slow, Short, Long, Good, Normal, Bad}

The Rule base for the example consists of 12 rules (we will refer to the rules by the number given):

```
1.  IF(Time is slow)AND(Path Length is Long)AND(Behaviour is Bad)THEN(Trust Level is Low)
2.  IF(Time is slow)AND(Path Length is Long)AND(Behaviour is Normal)THEN(Trust Level is Low)
3.  IF(Time is slow)AND(Path Length is Short)AND(Behaviour is Bad)THEN(Trust Level is Low)
4.  IF(Time is Fast)AND(Path Length is Long)AND(Behaviour is Bad)THEN(Trust Level is Low)
5.  IF(Time is slow)AND(Path Length is Short)AND(Behaviour is Normal)THEN (Trust Level is Medium)
6.  IF(Time is slow)AND(Path Length is Short)AND(Behaviour is Good)THEN(Trust Level is Medium)
7.  IF(Time is slow)AND(Path Length is Long)AND(Behaviour is Good)THEN(Trust Level is Medium)
8.  IF(Time is Fast)AND(Path Length is Long)AND(Behaviour is normal)THEN(Trust Level is Medium)
9.  IF(Time is Fast)AND(Path Length is Long)AND(Behaviour is Good)THEN(Trust Level is Medium)
10. IF(Time is Fast)AND(Path Length is Short)AND(Behaviour is Bad)THEN(Trust Level is Medium)
11. IF(Time is Fast)AND(Path Length is Short)AND(Behaviour is Normal)THEN(Trust Level is Medium)
    IF(Time is Fast)AND(Path Length is Short)AND(Behaviour is Good)THEN(Trust Level is High
```

After defining the membership functions and the Rule base, now we will show the path taken by this agent into three scenarios with various output. The graphs that represent the scenario are on the following.

Case 1.

The agent travels using 2 hops, from the front door to the kitchen. Before migrating to the kitchen, the agent uses 40 seconds processing time at the front door, and is given 10 as the mark for behaviour from the range 10 to 100. The path taken by this agent is shown in figure 1.

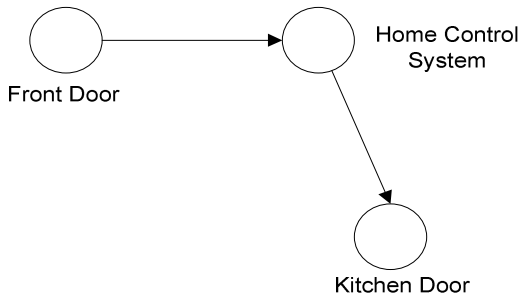

Figure 1. Guest agent travel path for Case 1

Case 2

The agent travels 5 hops, from the front door to the kitchen. Before migrating to the kitchen door, the agent uses 50 seconds processing time at the smart phone, and is given 80 as the mark for behaviour from the range 10 to 100, since it has interacted with Dieva the home owner. The path taken by this agent is shown in figure 2.

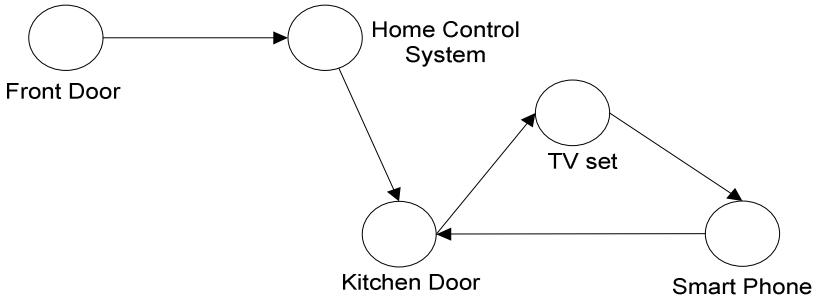

Figure 2. Guess agent travel path for Case 2

Case 3.

The agent travels 6 hops, from the front door to the kitchen. Before migrating to the kitchen door, the agent spends 60 seconds at the smart phone. Somehow, in this case, the agent is behaving differently. For an unknown reason, the agent travels to the Internet before going to the smart phone. Therefore, the mark for behaviour of this agent is reduced to 10. The path taken for this agent is shown in figure 3.

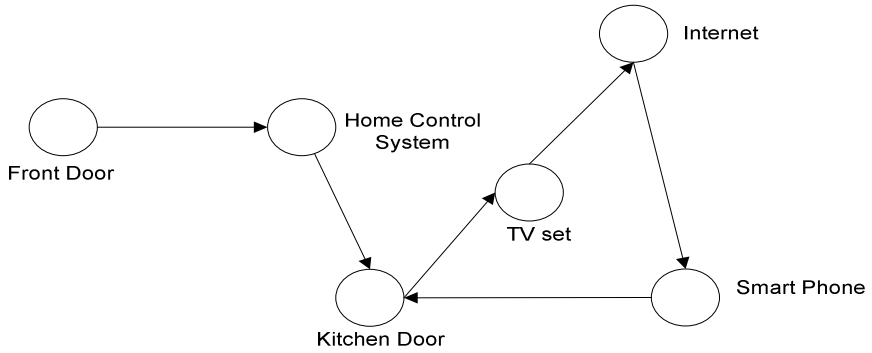

Figure 3. Guest agent travel path for Case 3

Using MATLAB 7.6.0 (2008R) fuzzy inference system and the Simulink, the crisp output for the decision making is defined by the following functions.

The Firing Strength for I $\in$ {Fast, Slow, Short, Long, Good, Normal, Bad} is defined by the S-function (1). The parameters k_i, w_i, μ_i, and σ_i are shown in Table 2.

Firing Strength $$FS_i(X) = e^{-k_i \cdot w_i \cdot (\frac{X-\mu_i}{\sigma_i})^2}$$ ..... (1).

For example, $$FS_{Fast}(X) = e^{-k_{Fast} \cdot w_{Fast} \cdot (\frac{X-\mu_{Fast}}{\sigma_{Fast}})^2}$$

The fuzzy output takes into consideration the decision factors, namely, time T, path length P, and behaviour B.

$$FO_m(T,P,B) = FS_g(T) \times FS_h(P) \times FS_i(B) \times \mu_j$$ ..... (2).

Here *g* $\in$ *{ Fast, Slow}, h* $\in$ *{Long, Short}, i* $\in$ *{ Good, Normal, Bad}, j* $\in$ *{ High, Medium, Low },* and *m* $\in$ *{ Rule1, Rule2, … Rule12 }.*

For example, $$FO_{Rule1}(T,P,B) = FS_{Slow}(T) \times FS_{Long}(P) \times FS_{Bad}(B) \times \mu_{Low}$$

The crisp output for the Trust Level is calculated by dividing the total Fuzzy Output by he total firing strength from each rule. The formula for calculating the Crisp Output is therefore:

Crisp Output $$CO(X) = \frac{\sum_{m=rule1}^{Rule12} FO_m(T,P,B)}{\sum_{m=rule1}^{Rule12} FS_m(T,P,B)}$$ ..... (3).

The output for each case is calculated using the above equations with the parameters from Table 2. For Case 1 and Case 3, the crisp outputs are 26.0, and 2.3, which fall under the Low trust level. Therefore, the authentication processes for Case 1 and 3 is sophisticated, and the guest agent is only allowed to retrieve non-crucial data, using limited memory. The guest agent is executed in a sandbox, which does not permit API calls to override the biometric lock of the kitchen door.

For Case 2, the crisp output is 43.0, which falls under the Medium trust level. Therefore, the guest agent in this case is allowed more resources, and has to perform a normal authentication process. The guest agent is not sandboxed and can therefore make the API call that overrides the biometric door lock.

5. CONCLUSION AND FUTURE WORK

The central security concern in a system where agents roam is on how to establish trust in the agent. The more an agent can be trusted, the fewer protection mechanisms need to be enforced. Therefore we provide an overview of mechanisms designed to protect mobile agent platforms from malicious agents. Then we raise the question of how to decide which mechanism or which combination of mechanisms is the most appropriate for a given situation. We argue that to be able to make an appropriate decision it is necessary to gather the right information on past behaviour of the agent. We therefore give an overview of decision factors published in the literature. Finally we show by example how a fuzzy logic controller can be used with the relevant decision factors as input to decide on the appropriate trust level, which is then used to select the protection mechanisms. For future work we would like to work out the examples in more detail, building a full fledged simulation of a smart home that supports a variety of realistic usage scenarios.

References

[1] K. Kostiainen, O. Rantapuska, S. Moloney, U. Holmstrom, K. Karvonen, "Usable Access Control inside Home Network". World of Wireless, Mobile and Multimedia Networks, IEEE International Symposium, page(s):1-6, June 2007.

[2] S.K. Das, D,J Cook, A. Battacharya, E.O Heierman III, T.Y Lin, "The role of prediction algorithms in the MavHome smart home architecture". Wireless Communications, 9(6):77-84, Dec.2002.

[3] J-M. Seigneur, S. Farrell, C. Jensen, E. Gray, Y. Chen, "Towards security auto-configuration for smart appliances". In proceeding of the Smart Objects Conference, Grenoble, France, May 2003.

[4] W. Jansen and T. Karygiannis, “Mobile Agent Security,” NIST Special Publication 800-19, National Institute of Standard and Technology, 2000.

[5] M. Alfalayleh and L. Brankovic, “An Overview of Security Issues and Techniques in Mobile Agents”, The school of Electrical Engineering and Computer Science, The University of Newcastle.

[6] P, Lee and G. Necula, "Research on Proof-Carrying Code on Mobile-Code Security". In 24th Principles of programming language (POPL), Page(s): 106-119, Paris, France, Jan.1997. ACM. http://doi.acm.org/10.1145/263699.263712

[7] W. M. Farmer, J. D. Guttman, and V. Swarup, “Security for mobile agents: Authentication and state appraisal,” In Proceedings of the European Symposium on Research in Computer Security (ESORICS) LNCS 1146, page(s):118 – 130, Sep.1996. Springer

[8] M.J Grimley and Monroe, "Protecting the integrity of Agents". In ACM Magazine, B.D, 1999.

[9] D. Chess, B. Grosof, C.Harrison, D. Levine, C. Parris and G. Tsudik, “Itinerant Agents for Mobile Computing,” Technical Report, Oct. 1995, IBM T.J Watson Research Center, NY.

[10] J. J. Ordille, “When Agents Roam, Who Can You Trust?”. Proceedings of the Annual Conference on Emerging Technologies and Application in Communication, page(s):188-191, Portland, Oregon, May 1996. IEEE
[11] F. Bagci, H. Schick, W. Trumler, T ungerer, “The reflective mobile agent paradigm implemented in a smart office environment”, Personal and Ubiquitous Computing, 11(1):11-19, October 2006
[12] C. Cao, J. Lu, “A Path-History-Sensitive Access Control Model for Mobile Agent Environment”, Proceedings of the Third International Workshop on Mobile Distributed Computing (MDC) (ICDCSW’05), vol. 6, page(s): 660-663, June 2005.
[13] K. K Nguyen, B. Jaumard, "Distributed and scalable control plane for next generation routers: A case study of OSPF", In 33rd IEEE Conference, page(s): 464-471, 14-17 Oct.2008
[14] M. Ganesh (2006) *Introduction to FUZZY SETS and FUZZY LOGIC.* New Delhi: Prentice-Hall of India