Steven T. Wolfe, Sheikh I. Ahamed, and Mohammad Zulkernine, "A Trust Framework for Pervasive Computing Environments," *Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pp. 312-319, IEEE CS Press, Dubai, UAE, March 2006.

A Trust Framework for Pervasive Computing Environments

Steven T. Wolfe and Sheikh I. Ahamed
Marquette University
Wisconsin 53233, USA
{swolfe, iq}@mscs.mu.edu

Mohammad Zulkernine
Queen's University, Kingston
Ontario, Canada K7L 3N6
mzulker@cs.queensu.ca

Abstract

In this paper, we present a flexible, manageable, and configurable trust scheme for the security of pervasive computing applications. Our intention is to develop a trust framework capable of recognizing the difference between "pure" and "managed" ad hoc network structure, and capable of operating in either situation or a combination of both. Our trust framework will minimize the effects of malicious recommendations related to trust from other devices and have the capability to transfer security functionality from devices with limited computing resources to other secure and powerful devices. The presented framework allows administrators to customize trust-related settings in an effort to create a more secure and functional network. Within our framework, wireless devices are broken down into different categories based upon available resources and desired security functionalities. A device's categorization determines its security functionalities and interactions with neighboring devices. By using this flexible framework, administrators can customize devices based upon the network's environment.

Keywords. Trust framework, pervasive computing environments, trust framework for ad hoc networks, and pervasive security.

1. Introduction

Pervasive computing environments focus on integrating computing and communication technologies with the surrounding physical environment to simplify day to day user activities [1, 2]. In pervasive computing, numerous, casually accessible, often invisible, frequently mobile or embedded devices are connected to an increasingly pervasive network structure and collect information about the surrounding environment of pervasive computing using sensors [2, 3]. Pervasive computing can be used in different applications, such as hospitals, smart classroom, construction sites, stadiums, shopping malls, emergency services, convention centers, battlefields, and inhospitable physical environments etc. Within wireless networks, a subcategory exists involving temporary, dynamic networks whose composition changes frequently. These environments, called ad hoc networks, pose unique problems to developers, administrators and users. The wireless ad hoc networks [8] can be further broken down into "managed ad hoc networks" and "Pure ad hoc networks". Managed ad hoc networks involve situations where assumptions can be made regarding the deployment environment and presence of other devices. In these situations, an administrator can utilize this information prior to deployment. "Pure ad hoc" environments consist of situations where no knowledge of the network environment is assumed. These networks require no pre-configuration and usually hold all devices equivalent.

Pervasive network structures are commonly made of ad hoc networks of mobile portable devices. Mobile devices operating in a pervasive network structure are typically limited by battery life and computational power. Further, devices leave and enter independently producing a volatile network topology. This volatile nature prohibits the usage of more standardized approaches to an array of common problems such as computer security. A paramount security concern of pervasive applications and networks is ensuring network integrity and preventing malicious use. Essentially, this matter boils down to device authentication. In a traditional wired network or even an infrastructure wireless network, a single centralized device or group of devices is responsible for performing this functionality. For instance, Public Key Infrastructure (PKI) provides a method of authentication using identity certificates that are issued by a trusted certificate authority (CA) [14]. Due to the nature of pervasive computing environments, authentication schemes involving a centralized approach, like PKI, are not practical.

Recent approaches [5, 8, 9] for pervasive/ad hoc environments use reactionary distributed approach that shift the burden of authentication to the all or subset of the devices in the network. Distributed approaches can differ considerably amongst themselves. The Resurrecting Duckling Model forces the network to form a hierarchical master/slave pairing [11]. The slave receives authentication information from its master only. In contrast, most distributed approaches [8, 9] involve the device making a judgment based upon its observations and recommendations from neighboring devices. By communicating with another device, the device places a certain level of trust in that other device. Generally, devices allow, disallow or limit communication based upon the other device's "trust level". Depending upon the approach [6, 8, 9] trust is some combination of recommendation, the trust in the recommender, direct monitoring, and the importance of traffic, risk or merely the absence of malicious activity.

In this paper, we propose a flexible trust framework for pervasive computing environments that can be customized for a "purely ad hoc network", a "managed ad hoc network", or any environment in between. Our goal is to create a trust scheme capable of rapid deployment in purely ad hoc environments but also customizable for managed ad hoc environments. In addition, the framework will provide a mechanism for shifting analysis from limited devices to more powerful, more physically secure devices. This framework allows for a number of customizable options regarding trust.

Paper Organization. Sections 2 and 3 will outline the overview of the framework, delineate the categories of devices, and explain the mechanisms involved. The outline will be followed by an explanation of the trust calculation, the entities involved in the calculation, and a corresponding flow model. Section 4 provides network environments and adversary models that demonstrate the benefits of our work. A brief overview of our implementation immediately follows. Finally, we describe the development environment used and propose future work to extend our findings.

2. Trust Framework

2.1. Desired Attributes

The constraining environment of ad-hoc networks and the complications with authentication, as discussed in previous section make the following attributes (A1)-(A7) essential for a trust scheme.

A1) Low Memory Footprint/ Resource Usage: Mobile devices are typically constrained by battery life and computational power. Any trust service should make efficient use of resources to prevent unnecessary overhead.

A2) Distributed: Any approach towards a localized trust scheme must be, in some fashion, be distributed. The highly volatile nature of ad-hoc networks prevents a more traditional, centralized approach.

A3) No pre-configuration necessary: Pre-configuration such as setting shared keys in multiple devices is a timely process and should not be a requirement for deployment.

A4) Capability to Customize: An administrator can utilize the deployment environment to make the network more secure and functional. The trust scheme should allow for customization.

A5) Circumvent malicious recommendations: A recommendation-based trust scheme should contain processes to identify and minimize malicious recommendations.

A6) Prevent attack from variety of attack scenarios: A mobile ad-hoc trust scheme should perform reasonably well against adversary network models made of either a single device, a small number of devices, or a large number of devices.

A7) Recalculation: The trust scheme should involve a periodic update of trust values. Ideally, the time window between updates should be minimal to quickly respond to new threats.

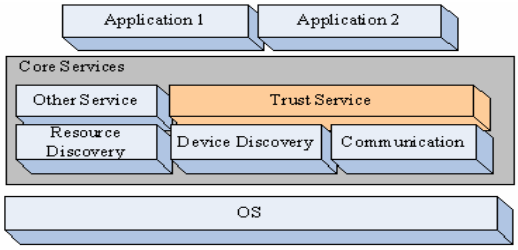

Figure 1. Trust service in the middleware MARKS [16]

2.2. Design Overview

The theory behind the trust framework is intended for usage at several layers. The prototype being developed has been designed to act as a thin client module residing in a group of core services. Core services are a group of essential services available to all applications. Our trust service utilizes a device discovery and communication service. This design is shown in Figure 1. Presently, our prototype is being developed as a service of MARKS: a middleware for pervasive computing of Ubicomp Research Lab. (www.mscs.mu.edu/~ubicomp) [16].

To allow access appropriately, the trust service will reside below applications and offer an interface to

allow applications retrieve a trust value and deterministically allow, disallow, or limit communication with the intended device. Within the service itself, several key components make up our trust service. These components are detailed in Figure 2. The user interface can be used by the administrator to customize the trust service to the existing network environment. The *rValue* manager estimates recommendation values for all devices that are used by trust calculation component to calculate trust along with the recommendations received from the local monitoring module. The request/response handler requests and replies to recommendation queries from neighboring devices. The application request handler replies to trust related queries from the pervasive computing applications.

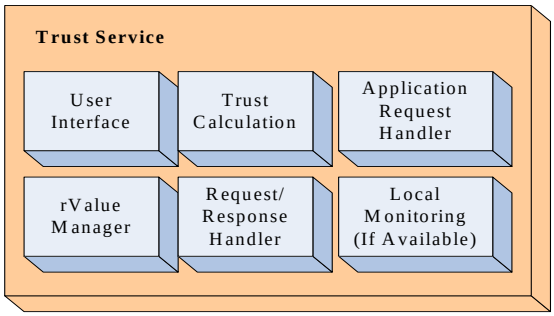

Figure 2. Trust Service Architecture

3. Management of Trust in the Framework

Our aim is to provide a configurable framework flexible enough to work efficiently in a pure ad-hoc environment but also allow pre-configuration to take advantage of known network characteristics. Additionally, the framework will provide a mechanism for the network devices where security functionality is minimal or undesired. In the following sections, we present a scheme for categorizing devices, calculating trust, and facilitating trust-related communication.

The framework allows administrators the ability to customize a network with any available knowledge. The more valid pre-configuration possible the more secure a network can be. This scheme provides for a more resilient network environment in a number of possible scenarios. When direct analysis is undesirable, other neighboring devices can provide this functionality and in doing so preserve battery life.

3.1. Device Categorization

Within the presented localized trust scheme, devices within a mobile network are divided into three categories. Categories are based on the desired amount of security functionality preformed by each device, available resources, and administrator preference. The three categories of devices in order of security services preformed are managerial, independent, and dependant. A brief summary of the categories is provided below.

3.1.1. Managerial Devices

Managerial devices are configured to perform the maximum amount of security services. These devices directly monitor network traffic, analyze and make network wide recommendations. Additionally, managerial devices can perform security services for neighboring devices. Most often, managerial devices would be able to perform security services for neighboring devices. In this scenario, recommendations from the managerial device would be given higher (or total) authority over recommendations from other devices.

3.1.2. Independent Devices

Independent devices perform security functionality only for themselves. Independent devices contain the ability to monitor proximal network traffic, analyze for suspected attacks and make recommendations to neighboring devices. However, these devices do not analyze network traffic for other devices and do not require a managerial device to analyze data. In terms of security, independent devices operate autonomously from neighboring devices.

3.1.3. Dependent Devices

Devices configured as dependant perform the most primitive security functionality. Dependant devices may perform minimal logging of network activity for further analysis by a neighbor. Additionally, these devices accept recommendations from neighbor devices. However, these devices do not perform any direct monitoring or analysis of network traffic. Most frequently, devices in this category are limited in computing power and battery life. To prioritize other processes, security functionality is delegated to one or more neighbor devices.

3.2. Quantifying Trust

Before proceeding into the calculation of trust, we will clearly define the word "trust" as it is used in this context. In our context, trust is the likelihood a device will not use network resources for malicious purposes. Abdul-Rahman and Hailes [6] provided a detailed explanation of trust for their model. We borrow the ideas that trust be subjective and conditionally

transitive from them. First, trust values are subjective. The trust value device *A* gives for device *B* may not be the same as the trust value *B* gives *A*. Further, trust is conditionally transitive. A device may recommend a trust value for another device. However, this recommendation must be taken along with the trust placed in its recommender. Once a trust value has been calculated, it is the responsibility of the application to decide what action to take.

In order for any model to work effectively, it will require some method for a device to decisively allow, disallow or limit communication with a neighboring device. This is implemented by assigning a numeric value from one device to another. This numeric value will represent how much the device "trusts" the other device. The numeric value will range from -1 to 1. A completely trusted device will be given a value of 1 and conversely a distrusted device will have a value of -1. The calculation of this value will rely on direct monitoring when available and recommendations of neighboring devices. Additionally, the true value of a recommendation will include the trust in its recommender.

3.3. Trust Value of Recommender

A recommendation taken at face value is a somewhat naive approach. Even in purely ad-hoc environments, a device should place more faith in recommendations of trusted devices as opposed to relatively new devices or known malicious devices. In any event, the recommendation level should be assessed in combination with the amount of trust placed in its recommender.

Our scheme calculates a weighted value of a recommendation based upon the trust placed in the recommender by the device. The recommendation value for a device represents the weight of a recommendation from that particular device. This level will be initially set for each device and will also provide for manual configuration and dynamic configuration based upon a history of established legitimate communication. The recommendation value (*rValue*) for a device will be a numeric value from 0 to 1. A device with a recommendation value of 1 will be highly trusted and its recommendation will be most influential. Conversely, a device with a recommendation value of 0 will be disregarded.

3.3.1. Guidelines for Setting *rValue*

Configurable *rValue*'s provide one of several opportunities for an administrator to customize a network with prior knowledge. By weighting recommendations, the administrator makes the network more resilient against attacks from malicious devices. Additionally, this recommendation scheme allows for devices incapable of direct monitoring. However, the ability to weight recommendations brings about the possibility of a highly trusted malicious device. In this event, a malicious device would be more empowered than in a traditional trust scheme. This possibility requires that an elevated *rValue* be given carefully. In addition to knowledge regarding the networking environment, the administrator should also take into account the physical security of the device.

3.4. Calculating Trust

The calculation of a device's trust value will be a combination of direct monitoring, recommended trust values, and the *rValue* of their recommenders. In the event of direct evidence of malicious use, a device's trust value should be determined solely by the device itself. The significance of this evidence is left as a customizable property.

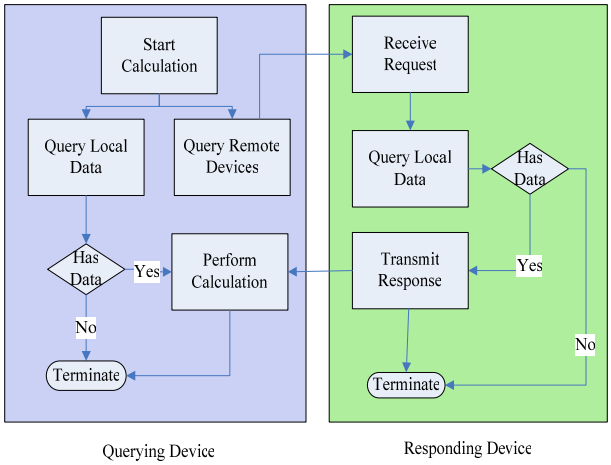

Figure 3. Calculation Flow Chart

A more complicated problem occurs when a device has no previous malicious knowledge of a device. In this event, the calculation of trust will come from a combination of recommendations from all willing and knowledgeable neighbors. A device will send out a request for recommendations for a device and calculate trust based upon the returned results. The formula for calculating trust will be as follows.

$$T_A(B) = \frac{(L_A(B) \cdot DF) + \left[RF \cdot \frac{\sum (T_{Di}(B) \cdot R_A(Di))}{\sum (R_A(Di))} \right]}{DF + RF}$$

$T_{Di}(B)$ = Trust Value device *Di* places on device *B*.

$L_A(B)$ = Monitoring value *A* places on *B*.

$R_A(Di)$ = Recommendation Value device *A* places on a recommendation from device *Di.*

RF = Recommendation Factor (significance of recommendations).

DF = Direct Factor (significance of direct evidence).

3.4.1. Trust Calculation Mechanism

The calculation mechanism occurs when a new device enters the network or at the time set for the next calculation of trust. At this moment, the device will begin the calculation mechanism. The flow of the process is detailed in Figure 3.

Initially, the device checks local monitoring data and queries remote machines for recommendations. The device waits a specified time for responses to come back. Upon receiving a recommendation request, a neighboring device will check its local monitoring data. If data is found, the machine will respond with that value otherwise it will not respond. The results of the recommendations are calculated using the formula presented earlier. If local monitoring data is found, this information is used in the trust calculation. The importance of local data and its effect on trust calculation should be high but is left to the administrator as a customizable option.

3.4.2. Example of Trust Calculation

Three devices (*D1*, *D2*, and *D3*) are deployed in a pervasive computing environment. *D1* calculates its value for *D3* in the following manner.

a) *D1* realizes it has no trust value for *D3*.

b) *D1* queries its nearby neighbors for recommendations.

c) Upon hearing *D1*'s request, *D2* responds with its trust value for *D3* (-0.3).

d) *D1* checks its monitoring data for *D3*. Finding none, the *rValue* will be 0.

e) *D1* calculates the trust value for *D3*.

$T_{D1}(D3) = ((0 * 1) + 1 * (0.3 * -0.3)/0.3)/2 = -0.15$

f) *D1* stores a trust value of -0.15 for *D3*. Any application communicating with *D3* will limit its functionality accordingly.

3.4.3. Interval of Trust Calculation

In order for a new device to communicate, surrounding devices must perform a trust calculation on the device prior to communication. Obviously, trust values should be recalculated periodically to ensure an accurate reading. Performing this calculation too frequently can lead to undesirable network overhead. Conversely, performing this calculation too seldom produces inaccurate trust values and allows malicious devices longer access to the network. The frequency of the calculation should be configured based upon network conditions.

3.5. Configurable Properties

The aim of this paper and supporting implementation is to provide a customizable

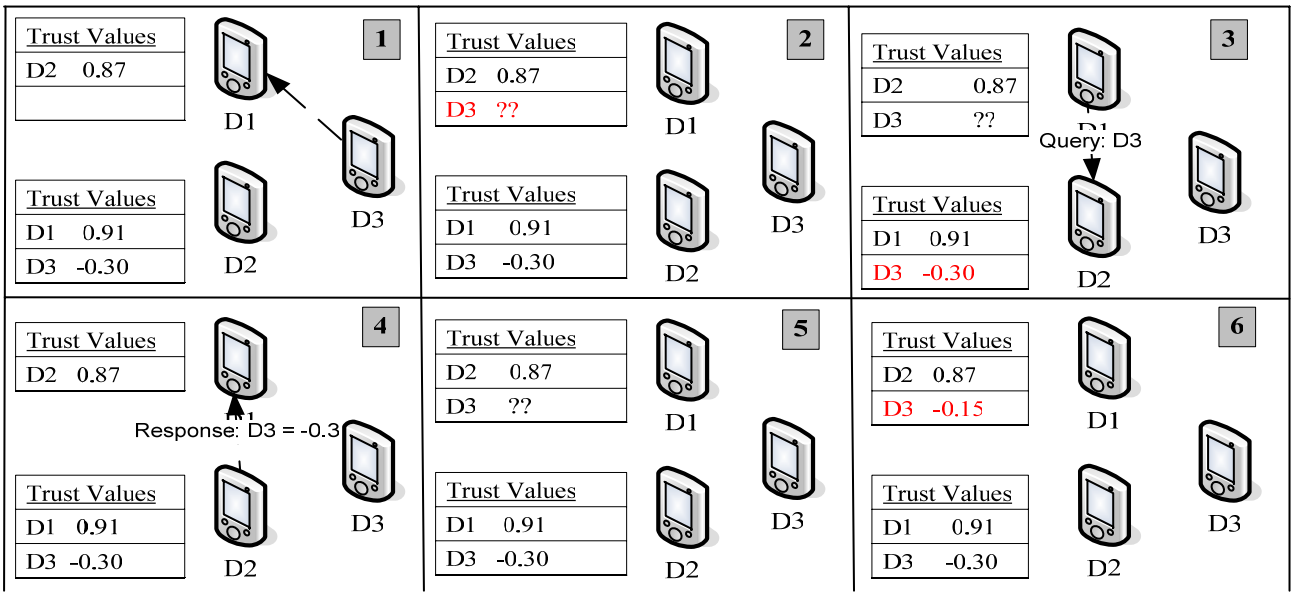

Figure 4. Trust Calculation

Property	Description	Possible Values	Initial Value
Device Categorization (DC)	The category of security functionality preformed	Managerial, Independent, Dependent	Independent
Recommendation Trust Value (rValue)	The trust value a node places in the recommendation from another node	0 (Insignificant) to 1 (Dominating)	0.5
Established Trust Update (ETU)	The ability to increase trust relationships due to established legitimate communication	Enabled, Disabled	Disabled
Trust Upper Limit (TUL)	The upper trust value a device can reach by established legitimate communication	Any numeric value from -1 to 1	0.5
Hierarchical Mode (HM)	Dependant node establishes a master/slave relationship with an independent or managerial node	Enabled, Disabled	Disabled
Trust Recalculation Time Period (TRTP)	The amount of time between calculating trust of neighboring devices	Any possible time value	60 seconds
Direct Factor (DF)	The significance of direct evidence in the trust calculation	Any numeric value from 0 to 1	0.9
Recommendation Factor (RF)	The significance of recommendations in the trust calculation	Any numeric value from 0 to 1	0.1
Initial Trust Value (ITV)	The trust value a device is assigned in the absence of recommendations or direct monitoring	Any numeric value from -1 to 1	0.5

Table 1. Customizable settings and default value

framework allowing administrators to customize the trust scheme to their needs. Table 1 provides a complete summary of the configurable options and a description of their effect on the trust scheme.

4. Example Applications

4.1. Numerous Malicious Devices

Network scenarios involving a large number of malicious devices create problems in ad hoc situations in schemes where trust calculation relies heavily upon recommendations. Any ad hoc scheme must have some mechanism for dealing with false recommendations. For instance, an administrator deploys three devices *(D1, D2,* and *D3)* to create an ad-hoc network for machines that may later come into the area. An adversary deploys three devices in an effort to make a coordinated attack against the existing devices. Attack scenarios involving large adversary models are not new to computer security. These scenarios pose an additional risk in trust schemes when recommendations, in the absence of previous evidence, are taken at face value. The specific nature of the attack is not addressed in this paper.

In our example, three malicious devices *(M1, M2,* and *M3)* are deployed in the geographical area and begin communicating with each other and devices *D1, D2,* and *D3.* When device D1 is asked to recalculate the trust value of its neighbors, the following events transpire.

a) For device *D2, D1* checks its direct monitoring for direct evidence against *D2.*

b) Finding no direct evidence, *D1* requests recommendations pertaining to *D2* from neighboring devices.

c) *D3* responds favorably to *D1's* request.

d) *M1, M2,* and *M3* respond negatively and report malicious recommendations in an attempt to halt communication between *D1* and *D2.*

e) After receiving the responses, *D1* concludes *D2* cannot be trusted and limits or halts communication.

In the preceding example, the three malicious devices have effectively used the trust mechanism as a weapon against the desired purpose of the network. A similar result could happen between *D1* and *D3,* and *D2* and *D3* effectively halting all legitimate communication. Our framework could resolve this problem or at least make the network more robust against this type of attack. Prior to deployment, the administrator can elevate the recommendations of *D1, D2,* and *D3* amongst each other. Other devices will be given the default recommendation value of the network. False recommendations from devices are taken at lower value than recommendation from *D1, D2,* and *D3.* Depending upon the level of elevation, the scheme could prevent against this type of attack from an adversary model much larger than the devices deployed.

4.2. Independent Devices

A further example consists of a network composed of numerous low-functioning devices. In certain environments, device limitations such as battery life and computational power constrain the amount of detection and analysis a device can perform. In this event, the typical isolated device approach may not be the best approach. Such scenarios could benefit from the ability to form a more structured relationship or to out source analysis to a more capable device. Stajano

and Anderson's Resurrecting Duckling Model [11] presents a solution where devices are associated with a master device. This solution embodies the scenarios we wish to describe. However, it is noted that the drawbacks of a mandatory association include complications following the loss of a master device and fairly extensive pre-configuration that may not be suitable for all environments. For this example, the administrator deploys five "low-functioning" devices *(N1 – N5)* in the presence of high functioning device that will act as a managerial device. The administrator has two options in our framework.

a) The administrator may directly associate *N1-N5* with the managerial device (*S1)*. *S1* will deterministically decide the acceptable communications of these devices. This option closely resembles [11].

b) The administrator may elevate the recommendation values for *S1* and other known devices. The trust decisions of *N1-N5* will be based solely on the recommendations of this group and, if any, the limited monitoring of the devices themselves.

5. Related work

Abdul-Rahman et al [6] has proposed a distributed trust model for managing trust. Their approach provides a quantitative scheme for trust in a distributed trust model using distributed recommendations. Within their model, each device maintains a discrete trust value for neighboring devices. The values proposed range from -1 (completely untrustworthy) to 4 (completely trusted). It requires trust to be transitive and devices make recommendations on another device's behalf. Recommendations are conditional based upon the recommending device's trust level. The trust value assigned by a device is computed formulaically based upon the recommended trust values and the trust values of their recommenders. Hence, it needs extra external processing support to deal with false or malicious recommendation. Moreover, it is not inherently designed to support the ad hoc network [8]. Thus it is not suitable for pervasive computing environments.

In Self-Securing Ad Hoc Networks [9], devices are trusted unless there is first-hand evidence or a sufficient number of neighboring devices stating otherwise. This research proposes a distributed authentication mechanism (A2) where authenticated devices posses a valid certificate. (A1)-(A7) are attributes of a trust scheme discussed in subsection 2.1. To obtain a certificate, a specified number of devices (k) must vouch for the validity of that device. This scheme hinges upon a localized trust scheme to determine whether to offer a partial certificate. In the absence of direct evidence or overwhelming recommendations from neighboring devices, a device will offer its partial certificate to any requesting device. Certificates are only temporary (A7) and will eventually need to be renewed. This recertification mechanism ensures a misbehaving device will not be allowed to remain connected. The recommended approach is targeted for purely ad hoc environments (A3) and requires that all devices perform monitoring and analysis.

Pirzada and McDonald [8] propose a distributed trust model (A2) based upon direct monitoring in combination with the utility and importance of the situation. In this scheme, trust values form a continuous range from -1 to 1 representing complete distrust to complete trust respectively. The solution separates the trust calculation into numerous trust categories. The addition of importance and utility is included to account for the spontaneous nature of ad-hoc networks. Pirzada and McDonald [8] explicitly contrast the differences between "managed" and "pure" ad hoc environments. Their solution is designed specifically for "pure" ad hoc environments, requiring no pre-configuration (A3). The calculation for trust contains a mechanism for devaluing the influence of malicious devices and recalculating perceived trust (A5, A7).

The work towards an intrusion detection system for pervasive computing environments [7] presents a foundation for extending an IDS to an ad hoc wireless environment. This proposal involves an agent-based approach to listen, identify and generate alerts of malicious activity. A static agent rests on each host and collects evidence of malicious activity. This evidence is transferred to a mobile agent who gathers the evidence, performs analysis and generates alerts if needed. Although closely coupled with a wired network, the approach functions when disconnected from the wired infrastructure [12].

Sun and Song [15] present a trust framework for an ad hoc networking environment based upon game theory and the distributed algorithmic mechanism design. The proposed trust scheme calculates trustworthiness on the reputation value of the device, the environment, time (A7) and additional quantities. In this model, reputation is composed of two parts: the action history of the device and recommendation of other devices. Sun and Song's trust framework requires that each device broadcast its trustworthiness to the network upon entering. This feature creates a vulnerability to malicious devices that misrepresent their trustworthiness to the network. Additionally, Sun and Song conclude that their proposal assumes malicious devices cannot work together.

Steven T. Wolfe, Sheikh I. Ahamed, and Mohammad Zulkernine, "A Trust Framework for Pervasive Computing Environments," *Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pp. 312-319, IEEE CS Press, Dubai, UAE, March 2006.

6. Conclusions and Future Work

Our aim for this paper is to present a trust framework that is built upon existing strategies for trust between the wireless devices in a pervasive setting. Recognizing the distinction between purely ad hoc and managed ad hoc, we have sought to develop a solution flexible enough for either. Additionally, our service contains functionality allowing for security analysis to be outsourced to other more powerful and physically secure devices. By enabling the administrator to customize the solution, the trust mechanism can be tailored to the environment of deployment. Further, customizable options are initially set to values ideal for a purely ad hoc environment. This feature allows for rapid deployment in pervasive computing environments. Moreover, it can be used in pervasive security tool for pervasive computing application.

We have implemented our trust framework on PDA's. The framework has been developed using C# and the Compact .NET framework. The devices deployed with the service are Dell Axim X50V PDA's. Presently, we are testing our prototype against a number of adversary networks models. These models include a single malicious device, several malicious devices, and environments where legitimate devices are outnumbered by malicious devices. As of now, we have calculated and tested the trust values for small networks. We will use a simulation tool called OMNet++ to determine the effectiveness and network overhead of our scheme in a large environment. Lastly, we will integrate this service into the group of core services of MARKS [16] in an effort to provide the basic pervasive computing functionalities to an application developer.

Acknowledgement

The authors would like to thank Pradeep Kannadiga for his helpful comments.

References

[1] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American*, vol. 265, pp. 66-75, Sept. 1991.

[2] S. K. S. Gupta, W. Lee, A. Purukayastha, and P. Srimani. (editorial). IEEE Personal Communications, *Special Issue on Pervasive Computing*, vol. 8, no. 4. pp. 8-9, August 2001.

[3] G. Abowd and E. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing," *ACM Trans. Computer Human Interaction*, vol. 7, No. 1, pp. 29-58, March 2000.

[4] S. Yau, F. Karim, Y. Wang, B. Wang, and S. K. S. Gupta "Reconfigurable context-sensitive middleware for pervasive computing," IEEE Pervasive Computing, 1(3), IEEE Computer Society Press, pp. 33-40, July-September 2002.

[5] S. Yi and R. Kravets. "Key Management for Heterogeneous Ad Hoc Wireless Networks," *Proceedings of the 10th IEEE International Conference on Network Protocols*, pp. 202-203, 2002.

[6] A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model," *Proceedings of the 1997 workshop on New security paradigms*, pp. 48-60, 1998.

[7] P. Kannadiga, M. Zulkernine, and S. Ahamed, "Towards an Intrusion Detection System for Pervasive Computing Environments," *Proceedings. of the International Conference on Information Technology (ITCC)*, pp. 277-282, IEEE CS Press, Las Vegas, Nevada, USA, April 2005.

[8] A. Pirzada and C. McDonald, "Establishing Trust in Pure Ad-hoc Networks," *Proceedings of the 27th conference on Australasian computer science*, vol. 26, pp. 47-54, 2004.

[9] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-securing Ad Hoc Wireless Networks," *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, pp. 567, 2002.

[10] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Support for Mobile Ad-Hoc Networks," *Proceedings of the 9th International Conference on Network Protocols (ICNP'01)*, pp. 251, 2001.

[11] F. Stajano and R. Anderson,"The Resurrecting Duckling Security Issues for Ad-hoc Wireless Networking," *Proceedings of the 7th International Workshop on Security Protocols*, 1999.

[12] The PanGo Networks Press Release, "PanGo Networks Launches New Technology for Proximity-Driven Mobile Internet Services," Feb. 2001, http://www.pangonetworks.com/release_3.htm.

[13] BizJournals, "Healthy Investment for hospitals: wireless nets," Sept. 2004, http://www.bizjournals.com/industries/health_care/hospitals/2004/09/20/eastbay_story7.html

[14] J. Weise, "Public Key Infrastructure Overview," *Sun BluePrints Online*, Aug. 2001, http://www.sun.com/blueprints/0801/publickey.pdf

[15] H. Sun and J. Song, "Strategyproof Trust Management in Wireless Ad Hoc Network", *Proceedings of the IEEE Canadian Conference on Computer and Electrical Engineering*, 2004.

[16] MARKS: a middleware for pervasive computing of Ubicomp Research Lab. (www.mscs.mu.edu/~ubicomp)