4-2008

An Impregnable Lightweight Device Discovery (ILDD) Model for the Pervasive Computing Environment of Enterprise Applications

Munirul H. Haque
Marquette University, md.haque@marquette.edu

Sheikh Iqbal Ahamed
Marquette University, sheikh.ahamed@marquette.edu

An Impregnable Lightweight Device Discovery (ILDD) Model for the Pervasive Computing Environment of Enterprise Applications

Munirul M. Haque
Purdue University, West Lafayette, WI, USA
Ubicomp Research Laboratory, Marquette University, Milwaukee, WI, USA
Sheikh I. Ahamed
Department of Mathematics, Statistics and Computer Science, Ubicomp Research Laboratory, Marquette University, Milwaukee, WI, USA

Abstract:

The worldwide use of handheld devices (personal digital assistants, cell phones, etc.) with wireless connectivity will reach 2.6 billion units this year and 4 billion by 2010. More specifically, these handheld devices have become an integral part of industrial applications. These devices form pervasive ad hoc wireless networks that aide in industry applications. However, pervasive computing is susceptible and vulnerable to malicious active and passive snoopers. This is due to the unavoidable interdevice dependency, as well as a common shared medium, very transitory connectivity, and the absence of a fixed trust infrastructure. In order to ensure security and privacy in the pervasive environment, we need a mechanism to maintain a list of valid devices that will help to prevent malicious devices from participating in any task. In this paper, we will show the feasibility of using a modified human- computer authentication protocol in order to prevent the malicious attacks of ad hoc networks in industrial applications. We will also present two separate models for both large and small networks, as well as several possible attack scenarios for each network.

SECTION I. Introduction

Due to the advancement of technology for low-cost pervasive devices and inexpensive but powerful wireless communication, pervasive computing is now becoming an integral part of our everyday life. Pervasive computing—the brain child of Weiser [3]—is now proving its feasibility in education, industry, hospitals and healthcare, battlefields, etc. Though the memory capacity of pervasive computing devices, such as personal digital assistants (PDAs) and smart phones has significantly increased in the last few years, it is still in its infancy when compared to the storage capacity of other memory devices used in distributed computing with a fixed infrastructure scenario. Along with limited memory capacity, low battery power places another constraint on pervasive devices, as pointed out in [23]. Another important aspect of the pervasive computing environment is the dependency of one device on another as well as the mutual cooperation of each device. In this volatile scenario, where each device can join and leave arbitrarily, the knowledge of current, valid neighbors is important for each device. If a device enrolls a malicious device as its valid neighbor, then the security and privacy of the whole system might collapse. Let us use a hospital as an example, where doctors, nurses, and other medical staff can communicate with one another through a wireless network. A hacker can easily enter the hospital premise as a visitor. He could then communicate with others posing as a doctor and access sophisticated and classified patient records. Hence, a valid device discovery protocol is very important for pervasive computing applications. This discovery protocol is responsible for updating devices with the latest information about current, valid neighbors, and thus, prevents any valid device in the network from communicating with a malicious user. To the best of our knowledge, such a protocol as a middleware service for pervasive computing applications is yet to be developed.

Although several papers [4]–[5][6] propose resource discovery protocols for dynamically changing environments, they do not address the impact of the protocol on issues like scalability, performance, and battery power optimization. For service discovery, some researchers have followed the insecure service discovery protocol in which each device in the network can access the services offered by other devices. Intentional naming system (INS)/Twine [7], INS [8], and service discovery in DEAPspace [9] are examples of this type of discovery protocol. Service location protocol (SLP) [10] and salutation consortium's protocol [11] take a conventional access control approach to ensure secure service discovery. Secure service discovery service (SSDS) [12] uses a trusted central server to provide several security features. Though device discovery is a part of service discovery, the main focus of these service discovery protocols is to make available services accessible while maintaining privacy and security. However, they do not address several active and passive attack scenarios.

Our paper differs from the usual issues handled by these service discovery protocols. Popovski et al. [13] discuss a protocol for device discovery in a pervasive computing scenario in which the unique address of devices is not identifiable. Devices have to concurrently listen on several frequency channels in order to discover others as mentioned by Bluetooth. Bluetooth wireless technology has been described in [14]–[15][16]. However, these protocols operate in the media access control (MAC) layer and are not suitable as middleware service. The "adaptability" issue that has not been considered by many of the device discovery protocols has been partially addressed in [17]. The main focus of universal plug and play (UPnP) [18], [19], Jini [20], [21], and open services gateway initiative (OSGI) [22] protocols is to overcome the barrier of heterogeneity in discovering devices. Our focus, however, is on maintaining a list of valid devices and building a model for discovering valid devices that is unassailable in the face of active and passive malicious attacks.

Hopper and Blum (HB) have shown the use of HB protocol [1], [2] in achieving a secure human–computer (HC) authentication, where the user has to communicate using a dumb terminal while in the presence of active and passive eavesdroppers. A variant of this is used in [27] for device authentication. In our proposed model, we adapted these protocols for device discovery, using an *ad hoc* scenario, in such a way so that the device discovery mechanism works in a distributed environment, where each device in the network can participate. At the same time, we introduced a set of new terms including "leader node" and ohm; that are discussed later. Our model has been designed in such a way that it incorporates security in every step. The incorporation of security from a design point of view and the ability to withstand attack are the unique features of this model. This model provides a second tier of security, while authentication builds the first tier. We are calling the model impregnable because it can withstand a wide variety of malicious attacks. It is also lightweight because it does not use a fixed infrastructure, and all the computations involve only binary and, xor, and rotate operations, which are very less expensive from a computational point of view.

The remainder of the paper is structured as follows. The licensed vocational nurse (LPN) problem is defined and explained in Section II. Our proposed model is delineated step by step in Section III. The characteristics and the ability of the model to withstand attack for large and small networks is shown in Sections IV and Sections V, respectively. Couples of crucial issues are elaborated in Section VI. Section VII shows the architecture and placement of impregnable lightweight device discovery (ILDD) in our developed middleware. Related works on device discovery approaches are discussed in Section VIII. Evaluation are presented in Section IX. Some open issues and our future plan have been placed in Section X.

SECTION II. LPN Problem

Given a random q ×nn binary matrix **A**, a noise parameter η ∈ (0,1/2), a random n-bit vector x, a vector v such that $|v| \leq \eta q$, and the product $z = \mathbf{A} \cdot x \oplus v$, find an n-bit vector x' such that $|\mathbf{A} \cdot x' \oplus z| \leq \eta q$ [1], [2].

A. Explanation of LPN

Let us take a scenario that is formed by a sender (S) and a receiver (R). Both of them share a common secret x of size n. S sends a challenge a, which is actually a string of arbitrary 0s and 1s of length n to R. Both S and R will calculate the Boolean inner product $a \cdot x$ that actually indicates the parity bit. The receiver will send the parity to the sender where S will check for the matching of the calculated parity.

Let us assume that the aforementioned procedure will be iterated for q rounds. At each round, a passive intruder can guess the parity bit response of R with a probability of 2−1. So the probability of guessing the parity bit response correctly in all q rounds is 2−q. But the problem here is that, if the procedure is continued for a minimum of nn times, by capturing nn challenge–response pairs, a passive intruder can easily compute the secret number x using the Gaussian elimination method.

In order to eliminate this problem, a noise term n has been introduced. It allows R to respond with an incorrect answer with probability n. R is permitted to send ηq rounds of improper response to the sender. Sender S will acknowledge R as a valid user if

$$\text{numberofincorrectresponses} \leq \eta q. \qquad (1)$$

SECTION III.

Overview of ILDD

Though we provide two separate models for small and large networks, the general steps are the same for both scenarios. At the end of ILDD, all the malicious devices are discarded, and each device is updated with a current valid neighbor list. Our model is specially suitable to large corporate or hospital environment where pervasive devices (PDAs, smart phones, cell phones, etc.) can communicate with one another through wireless connectivity. Since every device can listen to every other device in such a scenario, there is no headache of synchronization.

In our model, each device maintains a list known as a "valid neighbor list" that contains the nodes that have been declared valid by the leader node. The term "leader node" is elaborated in Section IV-A. Here, we are using the term "valid device" to denote those devices that are already in the valid neighbor list of others. This also indicates that these devices have passed the challenge—response-based device update mechanism in the previous phase. If a malicious device can make the necessary number of correct responses to pass the challenge—response phase, it will also be updated as a valid device in the valid neighbor list of other devices.

We have made a few assumptions that are as follows.

1. All the devices who have passed the authentication phase know a specific secret x of length nn. If there are malicious devices present in the network, they have bypassed the authentication phase, and hence, do not know the secret x.
2. All the authenticated devices know a special function $f(\boldsymbol{x})$ that has been used in generating a new secret x from the old one. As a result, even if a hacker gains the secret x at some stage, he will not be able to generate the new secret as $f(\boldsymbol{x})$ is not known. Since $f(\boldsymbol{x})$ has never been communicated through the transmission medium, there is no chance of it being accessed by a malicious user. Thus, we ensure the secrecy of x.

The process is initiated each time a device joins. In an idle scenario (lack of triggering due to joining), the system is triggered periodically. This initiation starts with the challenges sent by the leader node.

The *steps* in our approach are as follows.

1. Let us assume that there are Δ valid devices and μ denotes the total number of devices in the network. The variable μ encompasses both valid devices and devices that are waiting to be updated as a valid device. Let us consider the steps from the point of view of a specific valid device S.
2. The leader node will send a challenge to every other node in the network (this indicates that either a device is requesting to join the network or a specific amount of time has been elapsed without triggering ILDD). After receiving that challenge every device will generate the new secret x using $f(\boldsymbol{x})$ and use the new secret in responding to all challenges. This process will be elaborated in scenario 3 (Section IV-B).

3. Each valid device in the network will send all other devices in the network a random challenge $a \in 0,1^n$ 0,1n. So S will send arbitrary challenges a_1, a_2, a_3, … , $a_{\mu\text{-}1}$ to b_1, b_2, b_3, ……, $b_{\mu\text{-}1}$, respectively.
4. b_1, b_2, b_3, ……, $b_{\mu\text{-}1}$ will calculate ($a\cdot x$) modulo 2 and send their responses to S.
5. S will also calculate (aa. x) modulo 2 for all the challenges a_1, a_2, a_3,…, $a_{\mu\text{-}1}$, and compare the results with corresponding responses.
6. For all the matched results, S will send a "true" (1) recommendation to the leader node for the corresponding devices and a "false" (0) recommendation otherwise. So, from the point of view of the leader, a true recommendation indicates that the receiver node calculated the correct response while a false recommendation denotes an incorrect response.
7. Now steps 2)–5) will be performed for each of the valid devices (b_1, b_2, b_3,…, $b_{\Delta\text{-}1}$, b_Δ) present in the network.
8. After receiving all of the recommendations, the leader will calculate the validity of each device including itself, using (1), and then broadcast the valid result.
9. Every other device will update their valid neighbor list according to the leader's broadcasted results.

One thing to be noted: a device that is trying to update itself as a valid device, but not yet in the valid neighbor list of others, cannot send challenges to other devices. This device can only send responses in reply to challenges. At the same time, it cannot send a recommendation about any device to the leader. Even if this device sends challenges or recommendations, those will be ignored because the trusted nodes will not accept any challenge or recommendation from any node that is not in their trusted neighbor list.

When a leader node leaves the network, it selects another node in the network as the leader based on the criterions mentioned in Section IV-A and broadcasts a message about the new leader.

SECTION IV.

Model for a Large Network

A device that is already in the valid neighbor list will be updated as a valid device if

$$\mathbf{V} \geq \text{ceil}\big((1-\eta)\times k\big) - \Omega \qquad (2)$$

where **V** is the number of true recommendations from other devices to the leader about the device being validated, η is the maximum allowable percentage of noise (intentional incorrect answer), $\boldsymbol{k}$ is the total number of challenges received by one valid device. As each valid device receives challenges from every other valid devices except itself, **k** = Δ – 1, Ω is the expected number of malicious devices present in the network that have updated themselves in the valid neighbor list, and Δ is the total number of valid devices in the network.

So (2) can be rewritten as

$$\mathbf{V} \geq \text{ceil}\big((1-\eta)\times(\Delta-1)\big) - \Omega. \qquad (3)$$

A new device that has just joined the network but is not in the valid neighbor list of other devices will receive a challenge from every valid device in the network. Thus, it will actually receive Δ challenges. As a result, **k** in (2) has been replaced by Δ. So, a new device will be updated as a valid device if

$$\mathbf{V} \geq \text{ceil}\big((1-\eta)\times\Delta\big) - \Omega. \qquad (4)$$

As the output of the term $((1-\eta)\times\Delta)$ can be a float value we are taking the ceiling of the term. For example, if the output of $((1-\eta)\times\Delta)$ is 3.2, we take it as 4 because the number of recommendations is an integer value.

A. Characteristics of the Model (Large Network)

1. The protocol of Hopper and Blum is for HC authentication in which a user is communicating with the computer through an insecure channel. This protocol was not designed for an *ad hoc* environment in which the scenarios are quite different.

2. HB protocol is based on a single server–client scenario. However, in an *ad hoc* network, the scenario is completely different. An *ad hoc* network is formed by a handful of devices joining and leaving arbitrarily. Instead of providing a challenge from a single device/server, we modified the method so that each valid device sends only one challenge to every other node present in the network.

3. We have introduced the concept of a leader node that is chosen based on trust level. For the trust portion we used the trust model proposed in [36]. This model updates the trust level of each device dynamically, based on its interaction and behavior with other devices.

4. We made two significant changes in interpreting n, which has been defined as the probability of choosing an intentionally incorrect answer. a) Instead of using the term "probability," we redefined n as the maximum allowable percentage of noise. The purpose of this modification is explained in scenario 5 (Section IV-B). b) The definition of LPN says ηη∈ (0,1/2). But, in our approach, we cannot allow n to be near 1/2 that indicates almost 50% noise. If we allow a device to generate an intentionally wrong answer for about half of the times, then only around 50% of correct guesses will be enough for a malicious device to prove itself as a valid device. Thus, the probability of a malicious device to prove itself as a valid device reaches almost 1 irrespective of the number of challenges it receives as it can always guess a right answer with a probability of 0.5. The optimum value of n is an issue that needs further research.

5. A new term Ω has been incorporated; Ω denotes the expected number of malicious devices present in the valid neighbor list. The utility of the term is mentioned later in scenario 5 (Section IV-B).

B. How Does the Model withstand Attack

Here, we provide five possible attack scenarios that prove the effectiveness of our model. In the following scenarios, we consider a passive adversary as the one who can listen to messages but cannot alter the content. An active adversary, however, can change the content of a message.

Scenario 1 (Passive Malicious Device, Listening in the Network): Here, a passive malicious device listens to the challenge–response pairs and tries to regenerate the secret x. However, due to the introduction of noise, it is not possible for the adversary to regenerate the secret x even after listening to nn (assuming nn is the length of the secret) or more challenge–response pairs because it does not know which are the responses that are intentionally wrong.

Scenario 2 (Passive Malicious Device, Participating in the Validation): If a malicious device takes part in the validation phase, then it has to make correct guesses for a minimum of ceil((1− η) × **k**) – Ω challenges. Larger networks along with carefully chosen value of n and Ω make this nearly impossible.

Scenario 3 (Active Malicious Device): If an active adversary participates in the validation mechanism without prior knowledge about the secret x, then the situation resembles Scenario 2. However, an active hacker has the capability to modify the challenge or response. The hacker can use this capability in the following way: let us consider a scenario where the number of valid devices present in the network is 7 and the length of the secret x or challenge is 5. Thus, one device will receive six challenges from the other devices. The adversary will change these challenges to a specific challenge, for example, to 10 000. If the majority of responses of the devices are 0, the adversary considers 0 as the correct response for the challenge 10 000, otherwise 1 is considered. Now, it has a valid challenge—response pair. If it can generate five such challenge–response pairs, it will be able to regenerate the secret x using the Gaussian elimination method. Even then, the active hacker will not be able to utilize this information because we are generating a new secret in each phase, with the help of $f(\boldsymbol{x})$. Since this function is not known to the adversary, he will not be able to generate the new secret x from his knowledge of the previous secret $x \cdot f(\boldsymbol{x})$ works as follows. It generates y and z where

$$\begin{aligned} y &= \mathrm{ROL}(x, 1) \\ z &= \text{bitwise XOR of} x, y \\ \mathrm{ROL}(\mathbf{x}, \mathbf{1}) &= \text{Left rotate} x \text{by1bit.} \end{aligned}$$

Now, z becomes the new secret.

Scenario 4 (Handling Devices with Decreased Trust Level): The trust level of a device increases or decreases based on its behavior with other nodes in the network. If a device receives malevolent treatment from any other currently trusted device, it has the flexibility to send an intentionally “false” (0) recommendation to the leader about that malicious device irrespective of the response (correct or incorrect) it gets from that particular device. The definition of “malevolent treatment” and our plan to combat is mentioned in Section V. Thus, if a trusted device mistreats another device, its trust level will be decreased. Thus, this device will be removed from the network when the leader node receives the intentionally false recommendations from the mistreated devices.

Scenario 5 (Worst Case Scenario): Let us consider the scenario of a large network of 101 valid devices $b_{1,} b_{2,} b_{3,\ldots,} b_{98,} b_{99,} b_{100,} b_{101}$ where two of them (b_{100}, b_{101}) are malicious. This indicates that b_{100} and b_{101} have made the necessary number of correct guesses in the previous device update phase and joined the valid neighbor list. Let us assume that the value of $\eta = 10\%$. Thus, a device is permitted to respond up to 10 out of 100 challenges with intentional incorrect answers. So, b_1 has received 100 challenges from other devices and consider that it has sent ten intentionally generated wrong answers to ten valid devices that are within the range ($b_{2,} b_{3,\ldots,} b_{98,} b_{99}$). The remaining devices have received correct responses from b_1. Despite receiving correct responses, b_{100} and b_{101} can provide a “false” (0) recommendation about b_1 in an attempt to remove the network from b_1. Thus, the leader will receive 12 $(10 + 2)$ “false” recommendations and 88 “true” recommendations. As a result, this valid device will be discarded from the network according to (1). In order to handle this scenario, we have modified (1) by incorporating the term Ω The term Ω denotes the expected number of malicious devices present in the network that have made the necessary number of correct guesses to prove themselves as valid devices. By carefully choosing the value of Ω, we can avoid the aforementioned situation. For example, if we choose the value of Ω to be a minimum of 2 and use (3), then device b_1 will not be discarded in the aforementioned scenario. In essence, the value of Ω denotes the highest number of malicious devices that have been updated as valid, but still can be handled by our approach. This amount of malicious devices will not be able to have any negative impact on the validation of true nodes. Thus, Ω has an inversely proportional relationship with the number of valid nodes in the network. The greater the number of valid nodes, the greater the number of correct guesses a malicious device has to make to prove itself as valid. With the increase of valid devices in the network, the probability for a malicious device to pass the challenge—response phase decreases, and thus, enables us to choose a smaller value for Ω.

On the other hand, the introduction of Ω increases the probability of a malicious node to make the minimum number of correct guesses needed for validation by a bit. For example, in the aforementioned scenario, the minimum number of correct guesses that a malicious node has to make to prove itself as a valid device has been reduced from 90 to 88 as a result of introducing Ω. However, for a large network, this impact is quite negligible since making 88 correct guesses is almost equally impossible as making 90 correct guesses.

The redefined interpretation of η proves its utility in this example. If we interpret η as probability and set η= 0.1, it does not ensure the number of intentionally incorrect answers will not exceed 10% of the whole challenges. If a valid device gives more than 10 incorrect responses when η= 0.1, then the device will be discarded in the aforementioed scenario even after introducing Ω.

SECTION V. Model for Small Network

A device will be updated as a valid device if

$$V = k \quad (5)$$

where **V** is the number of true recommendations of other devices to the leader about this device and **k** is the total number of challenges received by the device.

For a device that is already in the valid neighbor list, **k** = Δ−1, whereas for a new device that has just joined in the network but is not in the valid neighbor list of others, the value of **k** will be Δ.

A. Characteristics of the Model (Small Network)

Along with the first three characteristics of the large-network model, this specialized model for small network has the following unique characteristics.

1. Here, we have omitted the introduction of noise (η) and Ω. Introduction of both η and Ω facilitates the malicious user with the increased probability of proving itself as a valid user by a bit. However, introducing η and Ω in a small network will have a much greater impact on the probability of a malicious user gaining access to the network. For a small-network scenario, where only a few devices are present, this increased probability is not acceptable. The purpose of $\eta$was to ensure the secrecy of x. Here, secrecy of x will be guaranteed by ensuring that the number of challenge–response pairs observed by the intruder is less than the length of the secret x. If the number of devices present in the network is μ and Δ denotes the number of valid devices, then

$$\begin{aligned} \delta &= \Delta\big((\Delta - 1) + (\mu - \Delta)\big) \\ &= \Delta(\mu - 1) \end{aligned} \quad (6)$$

where δ is the total number of challenge–response pairs generated, $\mu - \Delta$ is the number of devices present in the network that are not in the valid neighbor list but are taking part in the validation process.

Since every valid device sends a challenge to every other valid device as well as to devices that are taking part in the validation process, we get the aforementioned equation. In order to ensure the secrecy of x

$$\text{length of secret} x, n > \delta. \quad (7)$$

Thus, the passive intruder will not be able to regenerate x by capturing sample challenge–response pairs. As a result, the purpose of noise introduction becomes invalid.

2. The term Ω was used to overcome the false testimony against a correct response by devices that are malicious but present in the valid neighbor list. The term Ω has also been discarded for the same reason that n has been omitted. If a valid device is discarded due to the false recommendation of a malicious device, it can again join the network when the device update mechanism is again initiated, provided no

malicious devices have been able to make the necessary number of correct guesses during this phase. For example, if a malicious device that is not in the valid neighbor list makes the necessary number of correct guesses in phase i, it will be able to provide intentionally false recommendations about valid users in phase i+1. Here, phase i means the ith incident of the device list update mechanism. So, in phase i +1, a valid device may be trapped by the intentionally false recommendation of the valid but malicious device. It is very likely that this malicious device will be discarded in phase i +1 since it does not know the secret x. As a result, the temporarily discarded valid device will be able to rejoin the network in phase(i +2). On the other hand, even if we include a very small value of Ω, this will also benefit a malicious user with huge gain in its attempt to make the necessary number of correct guesses to pass the challenge—response phase. For example, consider a scenario where a malicious device is trying to prove itself as a valid device with $\eta = 0$ (because we omitted η), Δ= 5 (a small network where five valid devices are present), and Ω = 1; thus, (4) suggests that the malicious device will be declared as valid if

$$\begin{aligned} \mathbf{V} > \quad &= \text{ceil}\big((1-\eta)\times\Delta\big)-\Omega \\ > \quad &= \text{ceil}\big((1-0)\times 5\big)-1 \\ > \quad &= 4. \end{aligned}$$

But, if we omit Ω and use (5), then the malicious device will be declared as valid if V= 5.

This shows that correctly guessing 80% of the time is enough for the malicious device to prove itself as a valid device in the former case, whereas it has to guess 100% correctly in the later case.

This scenario also tells why we have not excluded the valid nodes from taking part in the authentication mechanism. If we do that, a malicious device that joined as a valid device by arbitrarily making correct number of guesses will remain in the network for eternity.

3. Here, the leader has the flexibility to dynamically change the length of the secret x according to the number of nodes present in the network. When the device discovery mechanism is initiated, the leader node knows what should be the minimum length of the secret x using (7). Based on this, it generates challenges of length $m = \delta + 1$.

The device discovery mechanism starts with challenges to everyone from the leader. Now, f(**x**) comes into play. It generates y and z where

$$\begin{aligned} y \quad &= \text{ROL}(x,1) \\ z \quad &= \text{bitwise XOR of} x, y \\ \text{ROL}(x,1) \quad &= \text{Left rotate} x \text{by} 1 \text{bit}. \end{aligned}$$

Considering the length of the challenge as mm and length of the secret x as n, there are following three possible cases.

1. $(m = n)$: z becomes the new secret.
2. $(m > n)$: We take the last $(m - n)$ bits of z and pad it at the front of the previous secret to form the new secret.
3. $(m < n)$: We take the last $(n - m)$ bits of z as the new secret.

As a result, the new secret x will always have the length $\delta + 1$ that is enough to ensure that no hacker will be able to regenerate x because the number of challenge–response pairs generated will be less than the length of the secret x.

This capability removes the burden of keeping a large enough secret that obeys (7) all the time.

B. How Does the Model withstand Attack

This model for a small network withstands first, second, third, and fourth attack scenarios depicted in Section IV-B. There is much less of a capability to prevent attack Scenario 2 for a small network since there are a fewer number of devices. This is the Achilles heel of a small network. The fifth attack scenario is not applicable for a small network because noise (η) is not present here.

SECTION VI. Critical Issues

A. Optimal Length of Secret x

In the small-network model, the leader node automatically varies the length of the secret x, depending on the number of nodes, to ensure security. However, for a large network, the scenario is quite different as the length of the secret is constant. We discussed the importance of inserting an intentionally false response or noise parameter (n) to maintain the secrecy of x. Unfortunately, this is not enough. If the number of valid challenge–response pairs generated is too high relative to η, then there is a possibility that all the n challenge–response pairs collected arbitrarily by the malicious node are valid (here a valid challenge—response pair means a pair where the response has not been intentionally altered). In that case, the malicious node will easily be able to reproduce the correct secret x. Considering $\eta = 0.2$ (Hopper and Blum [1] showed that this is one of the optimal values for η), Ω = 1, and the probability of choosing intentionally false answer is evenly distributed over the total number of responses sent, then the probability that an arbitrarily chosen challenge–response pair is valid becomes (1 – 0.2) = 0.8. Thus, the probability that a malicious node arbitrarily collects n challenge–response pairs and all of them are valid is 0.8n. This probability decreases as we increase the value of n, but at the same time incrementing for the value of nn increases the response time. We have seen the best performance at $n = 32$ by simulating (using OMNeT ++) an *ad hoc* network consisting of a large number of nodes, includes malicious nodes who are trying to regenerate the secret by arbitrarily capturing challenge–response pairs. The result of our experiment has been shown in Fig. 1.

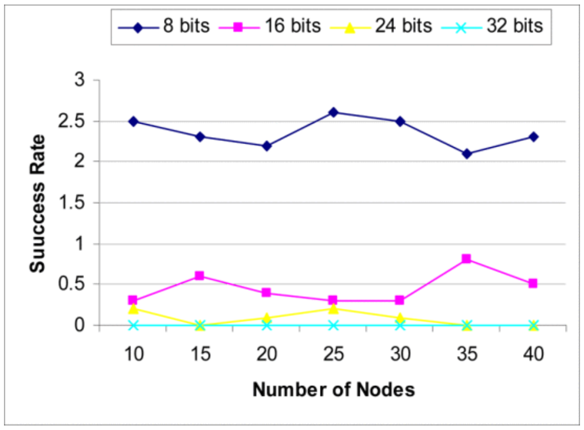


Fig. 1. Success rate in regenerating secret x through arbitrarily captured challenge–response pairs with increasing number of nodes.

We started with $n = 8$ bits. We ran the ILDD mechanism and collected n (here $n = 8$) arbitrary challenge–response pairs for 15 cycles. We then tried to regenerate the secret x using Gaussian elimination method. If all the collected challenge–response pairs are valid, the hacker can successfully regenerate the secret x through this procedure. We initiated the ILDD mechanism 15 times with different secret x (since secret x gets changed each time) and followed this strategy. Thus, the malicious node would have to regenerate the secret x by collecting nn arbitrary challenge–response pairs a total of 15×15=225 times. Then, we calculated the success rate in regenerating the secret. We continued this procedure for an increasing number of nodes. This whole procedure was then repeated for secret x with length 16, 24, and 32 bits.

In this experiment, we used three malicious nodes (though any number of malicious nodes can be used) with less than a total of 20 nodes in the network. When the total number of nodes became more than 20, we used five malicious nodes.

B. Optimal Value of Ω

In order to find an appropriate value of Ω, we simulated the model for a large network with increasing number of nodes starting from 5.

We started with $n = 32$ bits and $\eta = 0.2$. In this experiment, we used 15 malicious nodes (these nodes do not know the secret x and always attempt to break in as valid nodes by answering the challenges arbitrarily). The choice of 15 malicious nodes was arbitrary and any number of malicious nodes could have been used. These nodes request for joining the network and get challenge from the existing nodes of the network. If the malicious devices can guess the necessary number of correct responses, they would be declared as valid node. We used 15 different secret x, and with a specific secret, we ran the ILDD mechanism for large-network model for 15 times. So, the number of attempts in authentication by each malicious node will be a total of 15 × 15 = 225 times. We collected the maximum number of malicious nodes that were able to join the network as valid nodes in a single iteration.

Fig. 2 demonstrates the maximum number of malicious nodes making correct guesses with varying Ω in the large-network model. In our experiment, out of 15 malicious nodes, a maximum 4, 6, 10, and 13 nodes (with Ω = 0, 1, 2, and 3, respectively) were able to make necessary number of correct guesses in a single iteration out of 225 iterations. As we can see from the graph, due to the introduction of Ω, malicious nodes get more advantage when the network is quite small but this fades out as the network grows. Since malicious nodes can arbitrarily guess and sometimes make the correct number of guesses to join the network as valid, the model may suffer from the Scenario 5 mentioned in Section IV-B in the absence of Ω (Ω =0). According to the experimental result, the maximum number of malicious nodes that can be present in the network is 1 with the number of valid nodes more than 20 and Ω = 1. This means that, if Δ > 20 and Ω = 1, the model can effectively handle the presence of maximum one malicious node. So, if we take Ω = 1, we can run the large-network model when Δ > 20. According to the graph, no more than two malicious devices can be present in the network when Δ > 25 and Ω = 2. So, in this case, we can use the large-network model when Δ > 25. The corresponding value with Ω = 3 is Δ > 27.

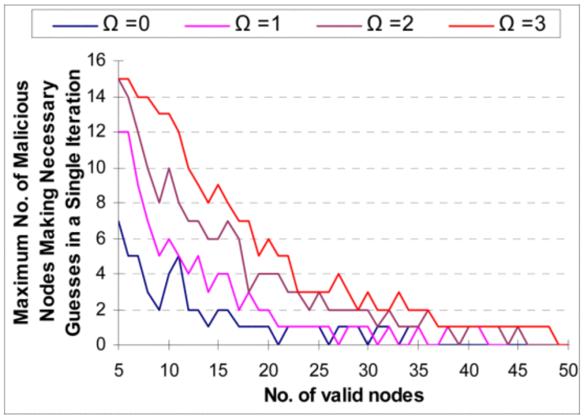


Fig. 2. Maximum number of malicious devices in a single iteration that made necessary number of correct guesses to join the network.

We would like to deploy the large-network model as early as possible, since it takes less time. Moreover, we need to ensure fewer numbers of malicious devices present in the network as valid as well. For example, from the aforementioned data, if we deploy the large-network model with Ω = 3 and Δ > 27, it ensures that no more than three malicious devices will be present in the network and the presence of up to three malicious device will not have any negative effect on the existing valid nodes. If we consider Ω = 1 and Δ > 20, we can restrict the maximum number of malicious device to 1. Moreover, Ω = 1 provides the earliest opportunity to deploy the large-network model. Considering all these, we chose Ω = 1.

We continued this experiment for 100 nodes. Since the maximum number of malicious nodes present in the network was always 0 from the moment Δ ≥ 48, the result for Δ >50 in the graph was not necessary.

C. Switching Between a Small and Large Network

As we propose two separate algorithms for a small and large network, an obvious question is when to switch from a small-network algorithm to large or vice versa. In order to answer this question, we deployed both models in the simulated environment using OMNeT++ and collected the required response time for the mechanism. In the case of a large network, the algorithm is a bit complex, but the length of the secret is constant over the number of nodes. On the other hand, for the small network, the model is simple but the length of the secret x increases exponentially with the number of nodes. Thus, we expected that the response time for small networks would increase rapidly as the network is expanded. Fig. 3 proves our assumption.

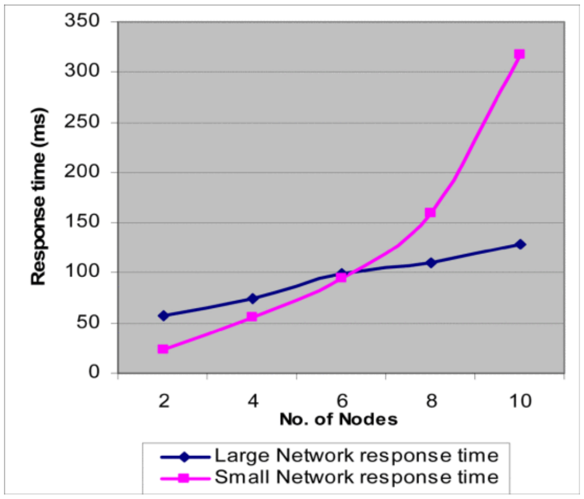


Fig. 3. Timing comparison between a large and small-network.

This experiment suggests 7 as the switching moment from a small to a large network and vice versa. But Section VI-B suggests that we can effectively introduce the large-network model when Δ > 20. So, we decided to switch from a small network to large when Δ > 20 and vice versa. This switching is transparent to the user.

Now another question arises. As a malicious node needs to guess a correct response to all the challenges in a small network, this model is quite rigid. The large-network model minimizes the time requirement but here a malicious node does not need to make a correct response to all challenges. As this research seems to be fine when the network has a large number of nodes, we assumed that it would work fine if we switch to a large-network model when the number of nodes reaches 21. So an issue of vulnerability arises as we just switch to a large-network model. To verify this issue, we simulated the following experiment.

We simulated the same scenario used in Section VI-B for a small- and large-network model. For the large-network model we used $\eta = 0.2$ and Ω = 1. Then, we calculated the maximum number of malicious nodes in the network that proved them as valid by making necessary number of correct guesses. We continued this procedure for an increasing number of nodes starting from 21 (since we are trying to see the effect just after switching).

Fig. 4 shows that the performance of the two models in terms of vulnerability is almost the same. The little spikes that denote the presence of one malicious node are handled by Ω.

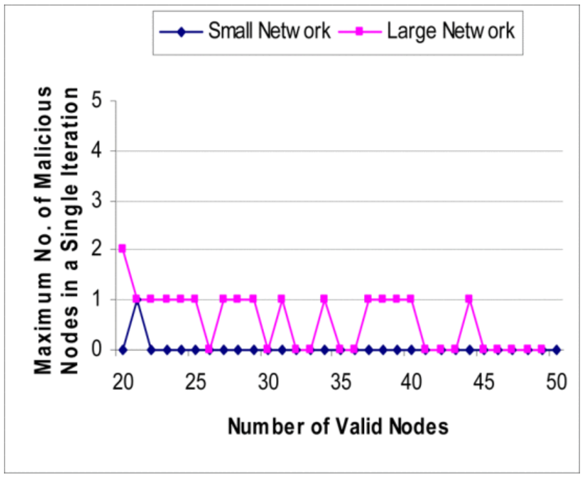


Fig. 4. Maximum number of malicious devices in the network by making necessary number of correct guesses in small- and large-network models.

D. Issue of the Leader Node

When there is no other node, the first node will be the leader node. When a valid node A sends the recommendation for all other nodes (B_i), it also includes the trust value that it has on the other nodes in the recommendation packet. We have discussed about this trust issue in detail in [36].

Later, the leader node generates the average trust value for every other node by dividing the summed trust value by the number of recommendations. Finally, when the leader node broadcasts the valid neighbor list, it also includes the IP address of the node that has the highest trust value. Let us assume that this node is M. Here, T denotes the “waiting time for the leader node” discussed in Section IX-C.

The leader node normally sends a notification packet before leaving and M becomes the new leader. If a new node requests to join the network but there is no response from the leader node in T milliseconds (which indicates that the leader node has left without notification), then M takes the position and starts sending challenge as the new leader.

E. Optimal Value of η

The hacker can introduce itself in the network in two ways: 1) by collecting n (length of the secret) valid challenge–response pairs and solving them using Gaussian elimination method to find the secret x and 2) by guessing the necessary number of correct responses.

We started with $n = 32$ bits and Ω = 1. In this experiment, we used 15 malicious nodes (these nodes do not know the secret x and always attempt to break in as valid nodes by following one of the strategies mentioned earlier). The choice of 15 malicious nodes was arbitrary, and any number of malicious nodes could have been used (we used 15 since, without loss of generalization, we can assume that the number of malicious nodes is much less than that of valid nodes).

Fig. 5 shows the result when these nodes request for joining the network and try to guess the necessary number of correct responses needed to be declared as valid node.

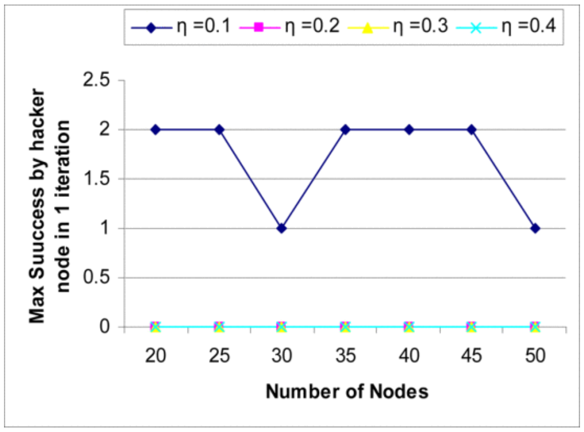


Fig. 5. Maximum number of malicious nodes present in one iteration when the first strategy is followed.

Fig. 6 shows the result when the hacker nodes just try to regenerate secret x by capturing n ($n = 32$) arbitrarily challenge–response pairs.

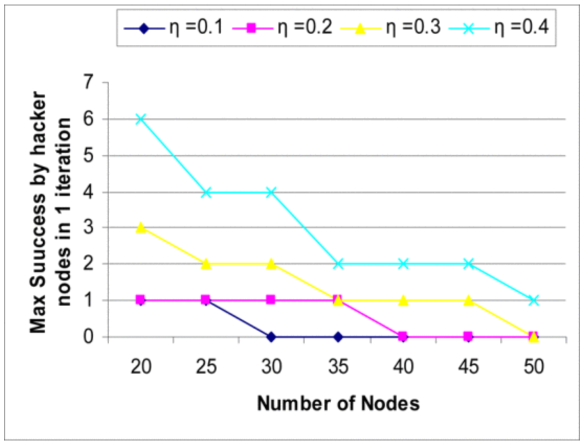


Fig. 6. Maximum number of malicious nodes present in one iteration when the second strategy is followed.

We used 15 different secret x, and with a specific secret, we ran the ILDD mechanism for large-network model for 15 times. So, the number of attempts in authentication by each malicious node will be a total of 15 × 15 = 225 times. We collected the maximum number of malicious nodes that were able to join the network as valid nodes in a single iteration.

In case of the first strategy, the probability of getting all valid challenge–response pairs out of n arbitrarily collected challenge–response pair decreases with the increased value of n. This is so because as we increase the value of n, the number of intentional incorrect answers in the network also increases. But this increased value of n helps the attacker if he adopts the second strategy since he has to make less number of correct guesses. As we increase the value of n, more and more malicious nodes make necessary number of correct guesses to prove them valid.

If we take $\eta = 0.1$, it gives good result against second strategy but not for the first one. But $\eta = 0.2$ gives acceptable result in both cases. In the second case, at most one malicious node was able to make the necessary number of correct guesses in one iteration when $\eta = 0.2$ and can handle the presence of one malicious node. So we have chosen 0.2 as the value of n.

SECTION VII. Architecture

Middleware adaptability for resource discovery, knowledge usability, and self-healing (MARKS) [28] provides the core communication facilities as well as other services, such as knowledge usability [29], secure, adaptive, fault-tolerant resource discovery (SAFE-RD) [30], and government emergency telecommunications service (GETS) self-healing [31]. ILDD has been added as a service to MARKS. The placement of ILDD service has been shown in Fig. 7.

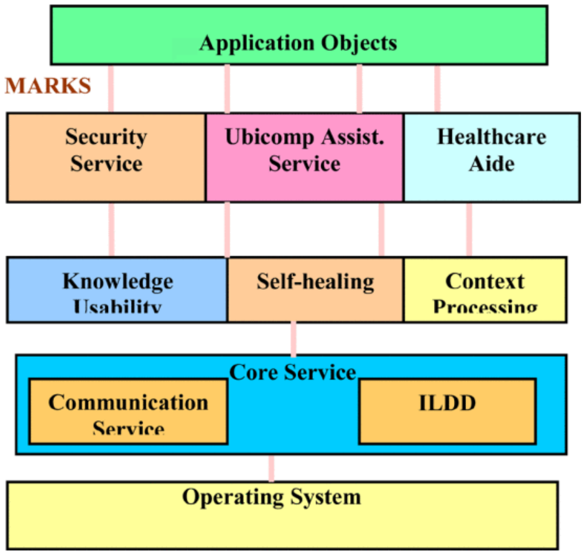


Fig. 7. MARKS architecture.

SECTION VIII. Related Work

Several device discovery protocols [13]–[14][15], [17], [26] have been implemented for *ad hoc* networks. These protocols are compatible with diverse wireless communication protocols like Bluetooth, 802.11, etc., but none of these protocols have the ability to serve as middleware service. These protocols work in the MAC and network layers and do not provide any application interface that is needed for application developers to use a middleware service.

The security of Web service has been discussed in [32]. However, the focus of this software system is to support the interoperability of applications over the Internet. Martin and Hung proposed a security policy [33] to handle different security-related issues including confidentiality, integrity, and availability for voice over Internet protocol (VoIP). However, our main focus is on secure service discovery in pervasive *ad hoc* networks. Aleksy *et al*. proposed a three-tier approach to overcome heterogeneity and interoperability issues [34]; unfortunately, their solution is appropriate for common object request broker architecture (CORBA) environment. Milosevic and Dromey proposed a model [35] for automating the behavior monitoring issues in contracts by employing two complementary methods. In our model, we have focused on making a validity decision based on behavior. Popovski *et al*. [13] have proposed a randomized distributed algorithm for device discovery with collision avoidance through the use of a stack algorithm, which fails to adapt itself with the dynamically changing environment of an *ad hoc* network. In UPnP protocol [18], [19], a controlled device publicizes its presence and services that it is willing to offer. Jini [20], [21] forms a community where each device can use the services provided by other devices in the community. OSGI [22] provides the framework for a home network that allows the user to communicate through devices with heterogeneous communication protocols, such as UPnP, Bluetooth, 802.11, etc. However, none of these technologies addresses the issue of power and resource optimization. The issue of maintaining a valid device list and corresponding security threats have not been discussed. Research studies in [14]–[15][16] discussed Bluetooth wireless communication. However, this protocol does not address power optimization and fault tolerance. At the same time, it does not focus on malicious attacks that we have discussed in our findings.

In a recent work [24], [25], the authors have shown that the Hopper–Blum protocol can be used to increase the security features of RFID. RFID reader sends several challenges to every RFID tag several times, receives the responses, and performs calculations. In our approach, the task of sending challenges is distributed to all valid nodes. Thus, the leader only sums up the recommendations of other devices. Due to the limitation of battery power, it is not possible for a specific node to handle all the challenges and calculations as the *ad hoc* network increases in size.

SECTION IX. Evaluation

This service can be evaluated in the following ways.

A. Implementation of the Prototype

We have developed a prototype of ILDD using the VC++. NET Compact framework on a set of wirelessly connected Dell Axim X50v PDAs in *ad hoc* mode. In Figs. 8–15, we provide some of the screenshots of our implementation.

B. Measurement of the Performance

Our model is a perfect example of the term "small memory footprint" that is a required characteristic for pervasive computing devices since these devices have much less memory storage capacity. Table I proves the fact.

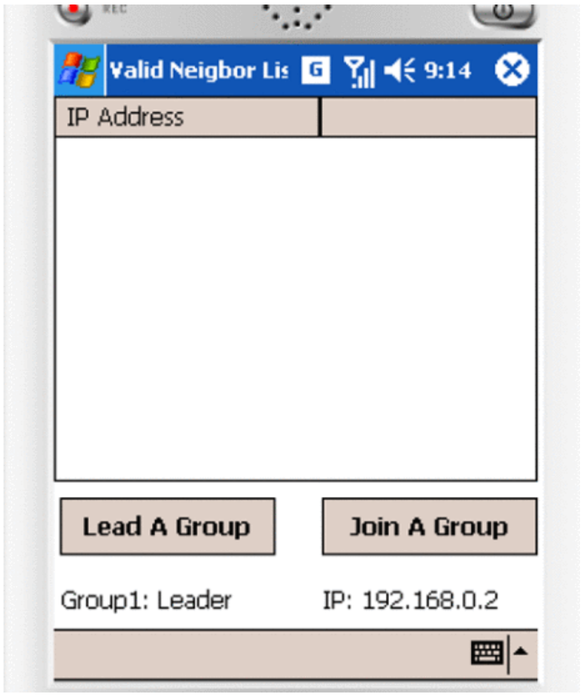
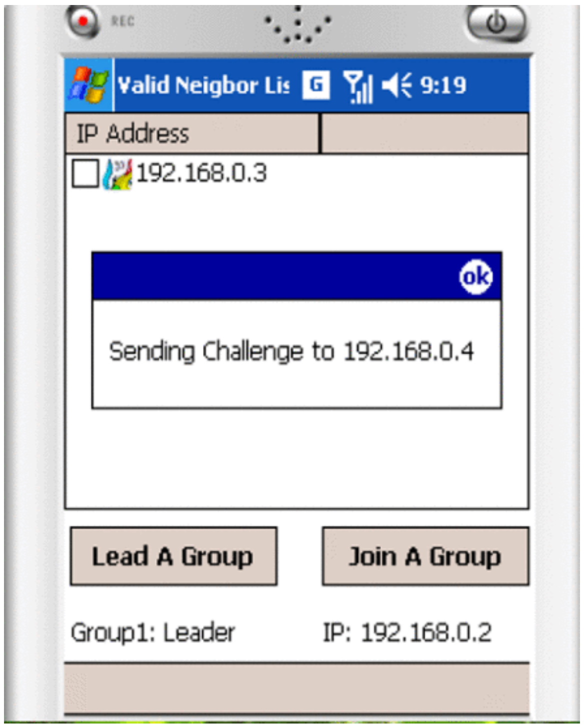


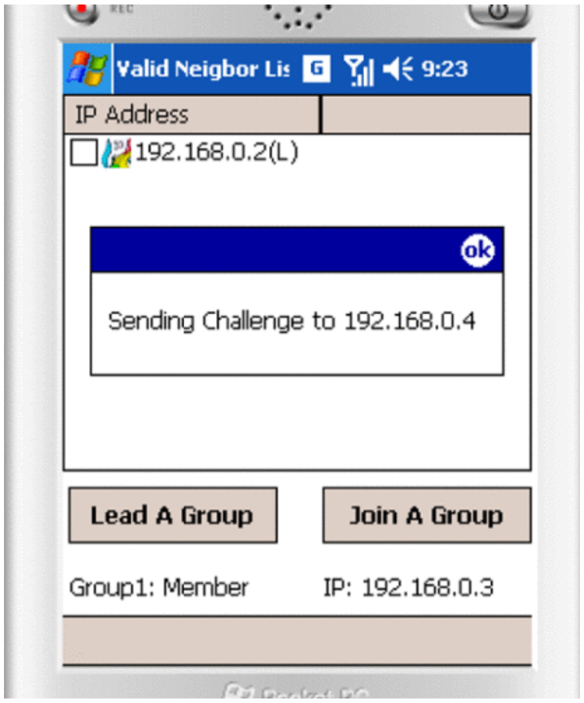
Fig. 8. Initial state of the leader.



Sending Challenge to 192.168.0.4

Group1: Leader IP: 192.168.0.2

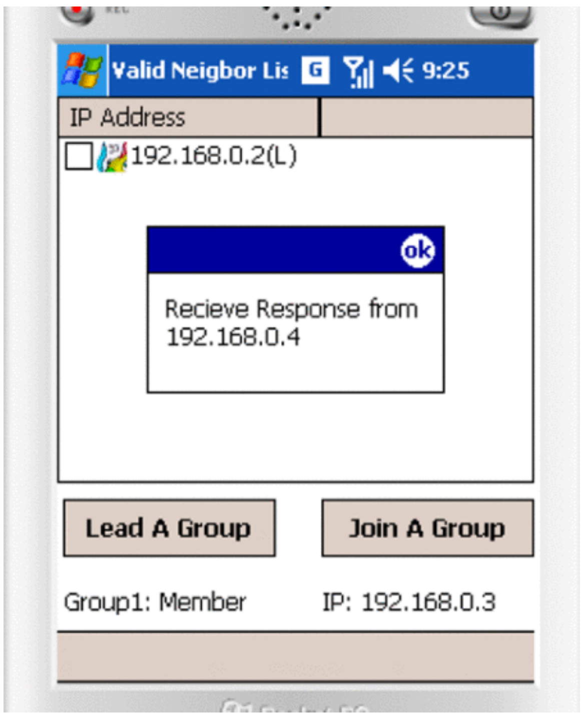
Fig. 9. Leader is sending challenge.



Sending Challenge to 192.168.0.4

Group1: Member IP: 192.168.0.3

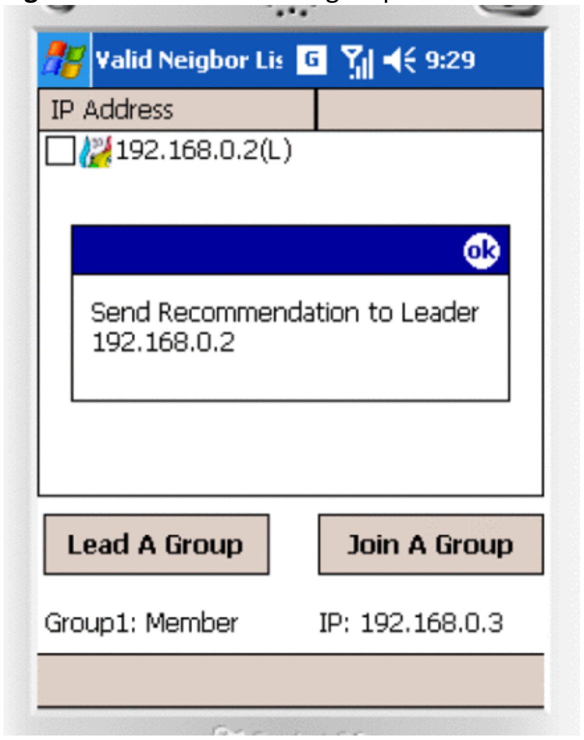
Fig. 10. Member is sending challenge.



Recieve Response from 192.168.0.4

Group1: Member IP: 192.168.0.3

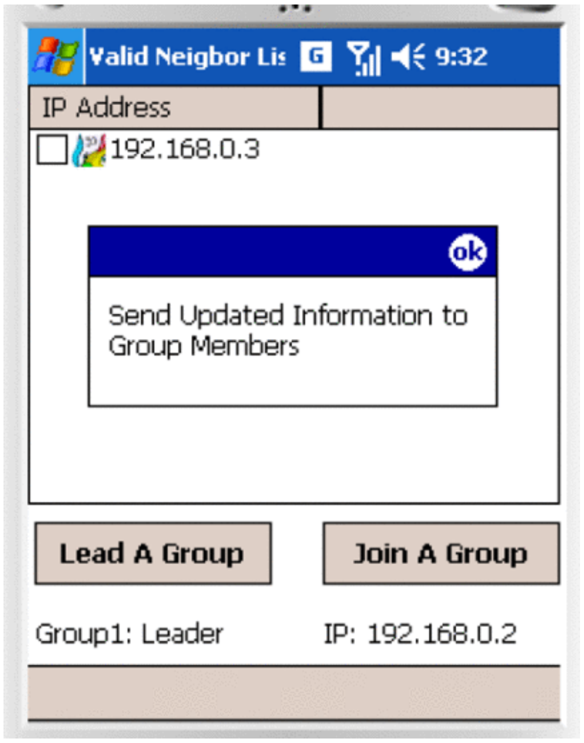
Fig. 11. Member is receiving response.



Send Recommendation to Leader 192.168.0.2

Group1: Member IP: 192.168.0.3

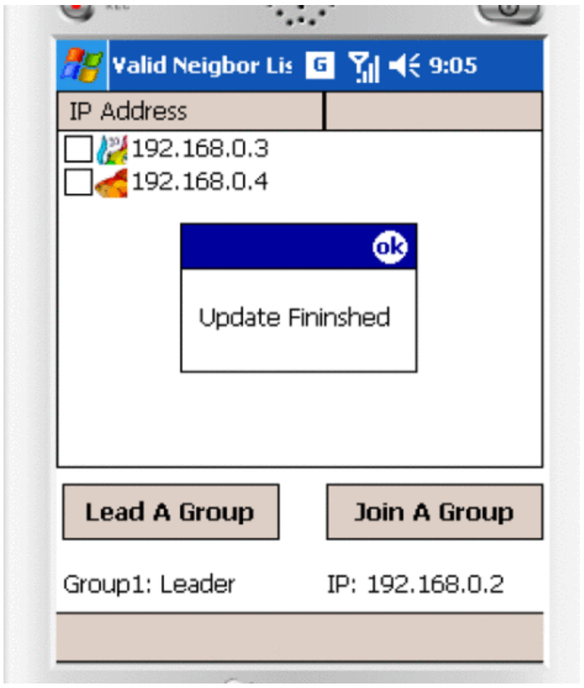
Fig. 12. Leader is receiving recommendation from member.



Send Updated Information to Group Members

Group1: Leader IP: 192.168.0.2

Fig. 13. Leader is sending updated neighbor list.



Update Fininshed

Group1: Leader IP: 192.168.0.2

Fig. 14. Updated neighbor list, leader.

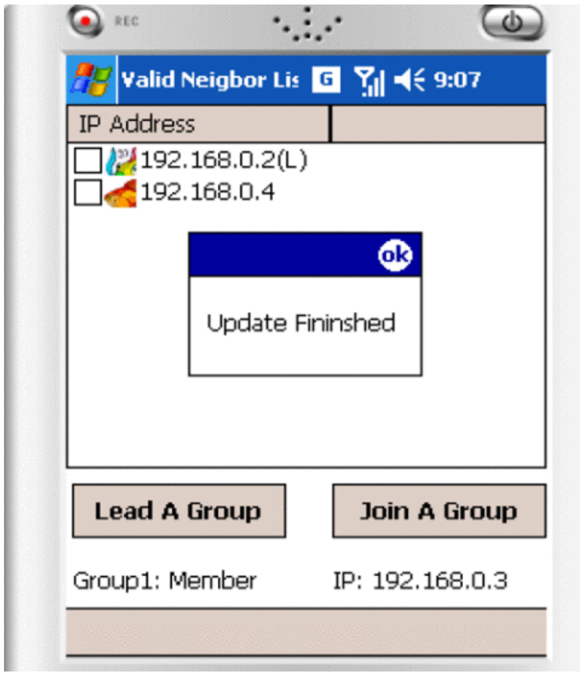


Fig. 15. Updated neighbor, member.

Our model consumes a small amount of battery power. In order to prove this statement, we measured the remaining amount of battery power in two scenarios: 1) when ILDD is not running on PDA and 2) when ILDD is running on PDA. We have modeled these two scenarios for two types of nodes: 1) normal valid node and 2) leader node. We did this because the work load of a normal valid node and a leader node is different. Figs. 16 and 17 show that the decrement rate of the curves indicating the remaining amount of battery power is almost the same in both cases. This actually indicates that even while ILDD is running, it is not consuming significant amount of battery power.

C. Performance Measurement Using Simulation

In order to check the scalability issue, we have simulated the ILDD mechanism using OMNeT ++. We tried to find out the time required for the ILDD mechanism when the number of nodes present in the network is rather large. Fig. 18 shows the time needed for completing an entire ILDD mechanism (completed all the steps mentioned in Section III) with an increasing number of nodes.

Table I Miniature Footprint of ILDD

	Line of code (LOC)	Executable file size
ILDD	1001	36 KB

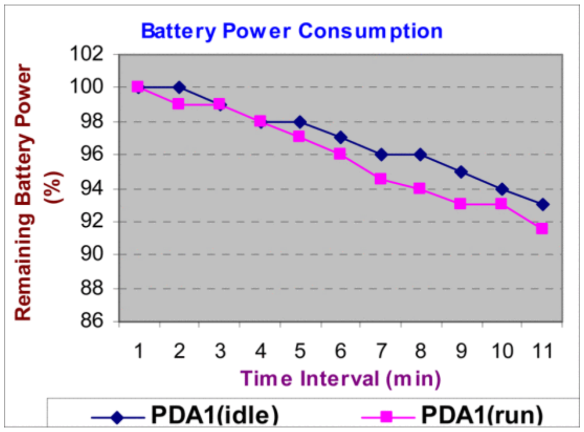


Fig. 16. Rate of decrement of remaining batter power before and after running the ILDD (normal valid node).

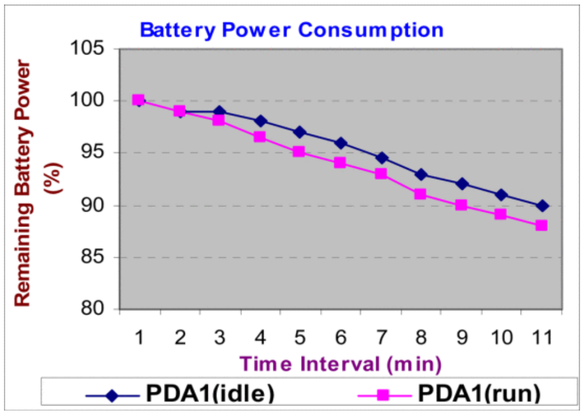


Fig. 17. Rate of decrement of remaining batter power before and after running the ILDD (leader node).

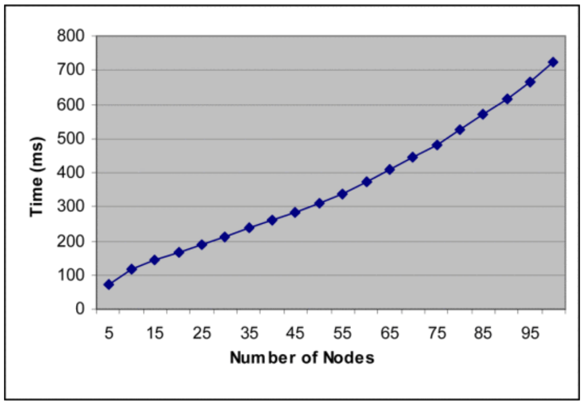


Fig. 18. Timing requirement for ILDD.

When we implemented the model, we used a timer in the leader node. When the timer expires, the leader node stops taking any more recommendation and starts the calculation.

By using the values we got from this experiment, we tried to formulate an equation that enables the leader node to determine the value of the timer. We used the equation given as

$$T = \begin{cases} t + \dfrac{N-5}{5} \times \Phi, & \text{where} N \geq 5 \\ t, & \text{otherwise} \end{cases}$$

where N is the number of nodes present in the network, t is the initial timing requirement, T is the waiting time for the leader node, and Φ is the average increase in the required time for an increase of every five nodes to finish the ILDD mechanism.

In our simulation, we started with five nodes and that takes 70 ms ($t = 70$ ms). We obtained this value by manually running ILDD in five PDAs. For any number of nodes up to 5, the leader node will wait for 70 ms for receiving all recommendations. From the simulated experiment, we found that $\Phi = 34.32$ (considering up to 100 nodes). So, we used $\Phi = 35$ in the equation. This issue will save the leader from waiting too long for all recommendations.

SECTION X. Conclusions and Future Works

In this paper, we have adopted the well-known Hopper–Blum algorithm in an *ad hoc* scenario and presented several pros and cons about its working methodology. As the number of nodes in an *ad hoc* network can vary substantially, we proposed two different models for both a small and large network. The unique characteristics of these models have been specified clearly. We also depicted several attack scenarios that can be handled with our model.

At present, we are working on a behavioral model that will be used to indicate the malevolent attitude of a node. This behavior model, though a simple one, takes into account issues like number of total requests, number of requests for the same service, number of rejections for a specific service, number of total rejections, etc. This model will be fitted in our developed trust model, and the output of the behavior model will play an important role in calculating dynamic trust.

We have not concentrated on a couple of issues like challenges to a selective number of devices, loss of challenge or response due to collision, spoofing or replay attack, etc. Cryptographic analysis of $f(\boldsymbol{x})$ is also saved for our future work. We are planning to incorporate these issues as future work.

Table II contains the meaning of notations used.

Table II Definitions of Notations

Notation	Meaning
x	The secret key
n	Length of the challenge or response
_	Number of valid devices in the network
µ	Total number of devices in the network
-	Maximum allowable percentage of noise (Intentional incorrect answer)
k	Total number of challenges received by one valid device
Ω	Expected number of malicious devices present in the network that have updated themselves in the valid neighbor list
-	Total number of challenge – response pairs generated
T	Waiting time for the leader node
F	Average increase in the required time for an increase of every 5 nodes to finish the ILDD mechanism

References

1. N. Hopper and M. Blum, "A secure human computer authentication scheme", *Carnegie Mellon Univ.*, 2000.

2. N. J. Hopper and M. Blum, "Secure human identification protocols", *Proc. 7th Int. Conf. Theory Appl. Cryptol. Inf. Secur.: Adv. Cryptol. Adv. Cryptol. (ASIACRYPT).*, vol. 2248, pp. 52-66, 2001.

3. M. Weiser, "Some computer problems in ubiquitous computing", *Commun. ACM*, vol. 36, no. 7, pp. 75-84, Jul. 1993.

4. P. Eronen and P. Nikander, "Decentralized Jini security", *Netw. Distrib. Syst. Secur. Symp.*, 2001-Feb.

5. *Hewlett Packard CoolTown. (2008)*.

6. *UC Berkeley. (2008). The Ninja Project: Enabling internet scale services from arbitrarily small devices*.

7. M. Balazinska, H. Balakrishnan and D. Karger, "INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery", *Int. Conf. Pervasive Comput.*, 2002.

8. W. Adjie-Winoto, E. Schwartz, H. Balakrishnan and J. Lilley, "The design and implementation of an intentional naming system", *17th ACM Symp. Operating Syst. Principles (SOSP 1999)*.

9. M. Nidd, "Service discovery in DEAPspace", *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 39-45, Aug. 2001.

10. E. Guttman, C. Perkins, J. Veizades and M. Day, *(1999). Service location protocol Version 2*.

11. *The Salutation Consortium Inc. (1999). Salutation architecture specification*.

12. S. Czerwinski, B. Y. Zhao, T. Hodes, A. Joseph and R. Katz, "An architecture for a secure service discovery service", *5th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom 1999)*.

13. P. Popovski, T. Kozlova, L. Gavrilovska and R. Prasad, "Device discovery in short-range wireless ad hoc networks", *IEEE Netw.*, vol. 3, pp. 1361-1365, Oct. 2002.

14. G. V. Zaruba and V. Gupta, "Simplified Bluetooth device discovery—Analysis and simulation", *Proc. 37th Hawaii Int. Conf. Syst. Sci.*, pp. 307-315, 2004-Jan.

15. F. Ferraguto, G. Mambrini, A. Panconesi and C. Petrioli, "A new approach to device discovery and scatternet formation in Bluetooth networks", *Proc. 18th Int. Parallel Distrib. Process. Symp.*, pp. 221-228, 2004-Apr.

16. G. V. Zaruba and I. Chlamtac, "Accelerating Bluetooth inquiry for personal area networks", *Proc. IEEE Global Telecommun. Conf.*, vol. 2, pp. 702-706, 2003-Dec.

17. K. Sohrabi, J. Gao, V. Ailawadhi and G. J. Pottie, "Protocols for self-organization of a wireless sensor network", *IEEE Pers. Commun.*, vol. 7, no. 5, pp. 16-27, Oct. 2000.

18. *Universal Plug and Play Forum. (2008). About universal plug and playtechnology*.

19. *Universal Plug and Play. (2000 Jun.). Understanding universal plug and play: A white paper*.

20. *Sun Microsystems. (2008). Jini network technology*.

21. *Sun Microsystems. (2008). The community resource for Jini technology*.

22. P. Dobrev, D. Famolari, C. Kurzke and B. Miller, "Device and service discovery in home networks with OSGI", *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 86-92, Aug. 2002.

23. M. Satyanarayanan, "Fundamental challenges in mobilecomputing", *Proc. 15th ACM Symp. Principles Distrib. Comput.*, pp. 1-7, 1996-May.

24. S. A. Weis, "Security parallels between people and pervasive devices", *Proc. 3rd IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, pp. 105-109, 2005.

25. A. Juels and S. A. Weis, "Authenticating pervasive devices with human protocols", *25th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO 2005)*, 2005-Aug.-14--18.

26. A. Wils, F. Matthijs, Y. Berbers, T. Holvoet and K. De Vlaminck, "Device discovery in residential gateways", *IEEE Trans. Consum. Electron.*, vol. 48, no. 3, pp. 478-483, Aug. 2002.

27. M. Haque, S. I. Ahamed, H. Li and K. M. Asif, "An authentication based lightweight device discovery (ALDD) model for pervasive computing", *Proc. 31st Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC 2007)*, pp. 57-64.

28. M. Sharmin, S. Ahmed and S. I. Ahamed, "MARKS (middleware adaptability for resource discovery knowledge usability and self-healing) in pervasive computing environments", *Proc. 3rd Int. Conf. Inf. Technol.: New Gen.*, pp. 306-313, 2006-Apr.

29. S. Ahmed, M. Sharmin and S. I. Ahamed, "Knowledge usability and its characteristics for pervasive computing", *Proc. 2005 Int. Conf. Pervasive Syst. Comput. (PSC 2005)*, pp. 206-209.

30. M. Sharmin, S. Ahmed and S. I. Ahamed, "SAFE-RD (Secure adaptive fault tolerant and efficient resource discovery) in pervasive computing environments", *Proc. IEEE Int. Conf. Inf. Technol. (ITCC 2005)*, pp. 271-276.

31. S. Ahmed, M. Sharmin and S. I. Ahamed, "GETS (Generic efficient transparent and secured) self-healing service for pervasive computing application", *Int. J. Netw. Secur.*, vol. 4, no. 3, pp. 271-281, 2007.

32. B. Carminati, E. Ferrari and P. C. K. Hung, "Web services composition: A security perspective", *21st Int. Conf. Data Eng. (ICDE 2005) Int. Workshop Challenges Web Inf. Retrieval Integr. (WIRI) Conjunction*.

33. M. V. Martin and P. C. K. Hung, "Toward a security policy for VoIP applications", *18th Annu. Can. Conf. Electr. Comput. Eng. (CCECE 2005)*.

34. M. Aleksy, M. Schader and C. Tapper, "Interoperability and interchangeability of middleware components in a three-tier CORBA-environment-state of the art", *Proc. 3rd Int. Conf. Enterprise Distrib. Object Comput. (EDOC 1999)*, pp. 204-213.

35. Z. Milosevic and R. G. Dromey, "On expressing and monitoring behaviour in contracts", *Proc. 6th Int. Conf. Enterprise Distrib. Object Comput. (EDOC 2002)*, pp. 3-14.

36. M. Sharmin, S. Ahmed and S. I. Ahamed, "An adaptive lightweight trust reliant secure resource discovery for pervasive computing environments", *Proc. 4th Annu. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom 2006)*, pp. 258-263.