Towards A General System for Secure Device Pairing by Demonstration of Physical Proximity

Yasir Arfat Malkani, Dan Chalmers, Ian Wakeman and Lachhman Das Dhomeja

Department of Informatics, University of Sussex
BN1 9QJ, Brighton, UK
{y.a.malkani, d.chalmers, ianw, l.d.dhomeja}@sussex.ac.uk

Abstract. Co-location of devices is a useful basis for access control policies for ad-hoc connections, as physical security, visibility and social norms provide reassurances to the device owners and participants. There are various possible techniques for demonstrating co-location through physical interactions, which others have started to explore. In some cases these provide the basis for encryption, in others simply confirmation of presence. In all cases these techniques are dependent upon hardware capabilities, offer varying physical scope and levels of attack resistance, and require different levels of user attention and visible public action. Different trade-offs amongst these considerations are desired in different situations. In this paper we present a framework for negotiating such pairings. This facilitates device identification, matching of pairing techniques to requirements, chains of communication to bridge between devices of different capability and improved security by combining techniques where possible.

Keywords: Authentication, security, co-location, discovery, pairing.

1 Introduction

Devices offer services. Device owners are willing for the devices of other people in the same location to use their device's services. How can we prove that these devices are co-located? How can we choose the most appropriate method(s) to prove co-location? There have been many recent proposals to provide secure device pairing [1-8] all varying in their security against different attacks, the needed hardware capabilities and the necessary level of user attention. In a world of heterogeneous devices and requirements, we need mechanisms to allow automated selection of the best protocols without requiring the user to have an in-depth knowledge of the minutiae of the underlying technologies. In this paper, we describe such a mechanism.

As motivation, let us introduce Angela, who is working in a well reputed organization. She organizes a meeting with representatives of some customers to give them a confidential briefing about a new product that her company is launching in near future. The meeting is organized in a hotel equipped with modern smart devices, but which is unfamiliar to Angela. On the meeting day, Angela is getting late, so she leaves her office in hurry and forgets to print some important documents required during the meeting. When she reaches the hotel, she wants to pair her laptop with a nearby printer to print the documents, without having to gain special permissions on the hotel network or pass files to a receptionist. That she has been allowed into the room with the printer is sufficient credentials. Next she goes to the meeting room, where she wants to pair her laptop with the projector *securely*, since the presentation carries some sensitive data. In addition to preventing eavesdroppers on a connection expected to last for several hours, Angela's laptop selects a mechanism that allows her to demonstrate to the room that the data is coming from her laptop. After her meeting and before leaving, she needs to discuss a confidential issue with her boss. At this time, she wants to pair her Bluetooth enabled headset with her mobile phone. Finally when she finishes everything and needs to leave the hotel, she wants to provide the hotel with a signature stored on her work smart-ID card to use in authenticating their invoice.

The scenario presented above embodies common problems in pervasive computing of ad-hoc interactions with unfamiliar devices and institutions, but can also make use of physical presence. It gives rise to two major concerns regarding the pairing process. First is how Angela makes sure that no one else can modify or read the sensitive data sent to the various devices. This requires setting up of keys for encryption, but also correct device selection in an unfamiliar environment. Second, while pairing the devices she needs to discover which pairing processes can be applied in each situation. To the best of our knowledge, there is no any existing secure pairing system that best fits in all four situations of the scenario. For example accelerometer based techniques are not practical for large devices, in a large room with a roof mounted projector radio signal and close-range techniques are likely to fail. Where a choice of pairing techniques is available not all users will be able to judge which one is the best to use. Further, a pairing system must not increase the complexity and the cost of the devices by requiring expensive dedicated hardware in all devices, but should accommodate the existing capabilities of the pairing partners and should be flexible enough to accommodate future technologies. We believe that a general pairing infrastructure for smart spaces can improve the security and usability of the pairing process. Our proposal is an attempt to integrate pairing schemes in a single model that facilitates association of any pair of devices in several situations by using their common co-location capabilities, and also to relieve user from choosing between dozens of pairing schemes.

The proposed architecture consists of two functional components: co-location servers and devices. Devices register their capabilities with an easily found database stored on the co-location server. When two devices need to associate, the client can query the co-location server to discover and acquire the required information to initiate a secure pairing with the target device. Different interactions to demonstrate proximity are possible and the selection requires consideration of the level of proximity required, the ease with which the interaction can be mimicked by an impostor, the availability of matching sensors to work with, the longevity of the association, and the desirability of the interaction being public. Based on the information from the co-location server, both the client and resource mutually execute a common co-location protocol. This protocol will involve the generation of a key from interaction with the environment – a successful pairing will arise when matching keys are generated. The selected interactions will generate an appropriate key for the nature of the intended association.

2 Background

The problem of secure device pairing continues to be a very active area of research in pervasive computing environments. The issue got significant attention from many researchers, after Stajano and Anderson in their seminal paper [9] highlighted the challenges inherent in secure device association. Their work [9, 10] has been considered as the first effort towards secure transient association between devices in ubiquitous computing environments. They proposed a master-slave model which maps the relationships between devices. The pairing process is done by agreeing a secret key over the physical connection (such as using a cable). Though the secret key is transferred in plain-text and cryptographic methods are not used, it is susceptible to dictionary attacks. In reality, it is also difficult to have common physical interfaces in all the devices, and carrying cables might not be feasible all the times. Balfanz et al. [2] extended Stanjano and Anderson's work and proposed a two-phase authentication method for pairing of co-located devices using infrared as a location limited side channel. In their proposed solution, pre-authentication information is exchanged over the infrared channel and then the user switches to the common wireless channel. Slightly different variations, of Balfanz et al [2] approach, are proposed in [4, 6, 11, 12], which also use location limited side channel to transfer the pre-authentication data. The common problem with these approaches is twofold: first, they need some kind of interface (e.g. IrDA, laser, ultrasound, etc) for pre-authentication phase and are vulnerable to passive eavesdropping attack in the location limited side channels, e.g. two remotes and one projector. Some location limited side channels, such as infrared and laser, are highly vulnerable to denial of service (DoS) attack. Some other pairing schemes including Bluetooth require the human operator to put the communicating partners into discovery mode. After discovery and selection of a device, the channel is secured by entering the same PIN or password into both devices. Although it is a general approach, it gives rise to a number of usability and security issues [13, 14]. For example, a short password or PIN number makes it vulnerable to dictionary or exhaustive search attacks. Further, in Bluetooth pairing an adversary can eavesdrop to break the security from a long distance using powerful antennas.

Recently proposed schemes [1, 5, 7] use audio and/or visual channels for a secure pairing process. Seeing-is-Believing (SiB) [5] uses two dimensional bar codes for exchanging security relevant information between the devices; while the Loud and Clear [1] system exploits annunciated nonsensical sentences corresponding to a shared key. Both of the schemes suffer from a few problems, such as SiB requires that one of the peers must be equipped with camera; while in Loud and Clear a speaker is required. Camera equipped devices are usually prohibited in high security areas; while the latter is not suitable for hearing-impaired users. Further, bar code scanning requires sufficient proximity and light in SiB; while Loud and Clear places a burden on the user for comparison of audible sequences. An adversary can easily subvert bar code stickers on devices in SiB; while ambient noise makes authentication either weak or difficult in Loud and Clear scheme. Saxena et al. [7] extended the work of McCune et al. [5] and proposed a scheme, which requires one device to be equipped with a light detector or a camera and the other with a single LED. When the LED on the device blinks, the other device takes a video clip. Then, video clip is parsed to extract an authentication string. This scheme has many of the limitations as SiB, such as requiring close proximity and a camera. More recently, the idea of shaking the devices together to pair them has become more common. In this approach two devices are hold and shaken together simultaneously, common readings from the embedded accelerometers in the devices are used to pair them together. Smart-its-friends [15] was the first effort towards this approach. The follow-on method to Smart-its-friends is shake well before use [8]. Mayrhofer and Gellersen extended the Holmquist et al. approach and proposed two protocols to securely pair the devices. Both of the proposed protocols exploit the cryptographic primitives with accelerometer data analysis for secure device-to-device authentication. Shaking the devices together is always not possible, since there is large variety of devices, such as printers, projectors and laptops that can not be shaken.

In contrast to all of above approaches, Varshavsky et al. [3] proposed Amigo [3] system, which exploits the knowledge of common radio environment of communicating partners to securely pair the two co-located devices. Since Amigo exploits Diffie-Hellman key exchange method with the addition of a co-location verification stage, it is computationally not feasible for many devices in pervasive computing environments. Further, there may be many pervasive computing environments where wireless communication is not in use, where the radio data is not available to process or where the wireless network is easy to eavesdrop on while remaining hidden.

In summary, no one has yet devised the perfect pairing protocol. Pairing protocols vary in the strength of their security, the level of required user intervention, their susceptibility to environmental conditions and in the required physical capabilities of the devices. In the remainder of this paper, we show how different protocols can be integrated within a general architecture for proving co-location, which is sensitive to the trade-off amongst the identified strengths.

3 System Architecture

Figure 1 illustrates the high level architecture of the proposed system, and figure 3 shows a more detailed sequence diagram of communications used in the proposed system. Devices move between four states: *initialization and registration, device discovery, authentication and paired*. In the *registration* process, the device generates capability information to send to the co-location server. Thus, each device becomes a visible part of the system and can benefit from any other legitimate device in the system by creating an association with it. After *registration*, a client moves into the *discovery* state. The client searches for pairable devices in the vicinity during this state by querying the co-location server. The latter performs a match-making process based on the client's query. It produces communication and co-location capability information based on common capabilities of both client and matching device(s) (resources). The co-location server provides this information to both devices to smoothly derive the operations of subsequent *authentication* state.

Once the device enters the *authentication* state, the received information from the co-location server is used to execute a common authentication scheme. Finally, if the client is successfully authenticated, it enters the *paired* state. During the *paired* state, the client periodically enters an evaluation process, where the expiry condition of the given credentials is tested. Based on the outcome of the evaluation process, the client could either remain in a *paired* state or the given credentials are revoked. In the remaining part of this section, we will discuss the design details of our proposed system.

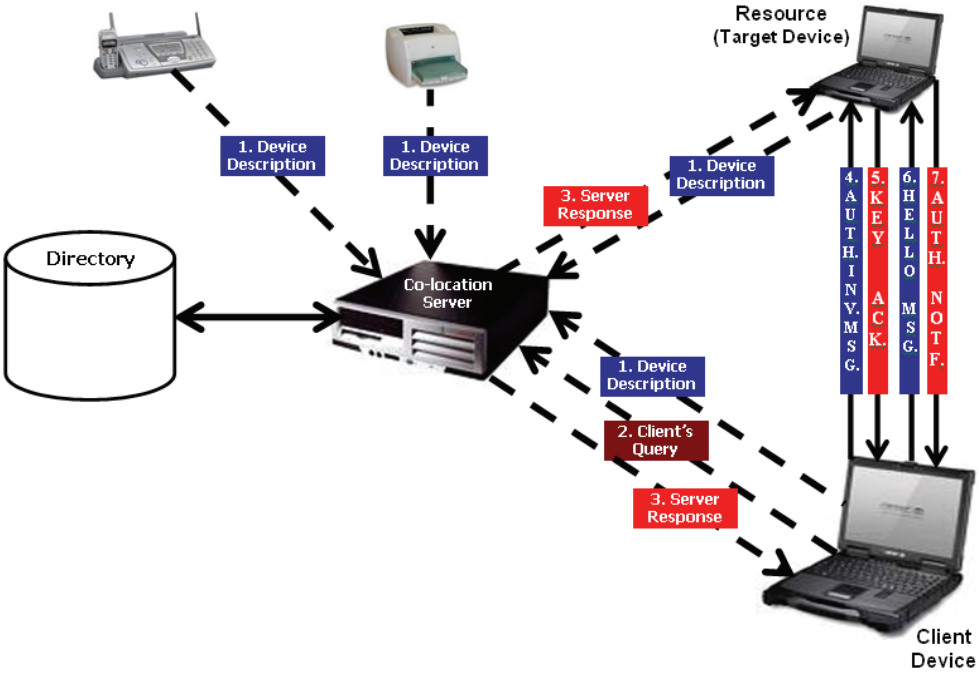

Fig. 1. High level architecture of the proposed system

3.1 Bootstrapping and Registration

Bootstrapping in our model refers to the system initialization and advertisement of co-location servers. Devices discover the co-location servers for registration by listening to a multicast address. A co-location server periodically multicasts its address, so that devices can find it and so register. During registration, the device component is responsible for providing its capabilities in XML form to the co-location server to store in the directory.

The co-location server might run with other local services (e.g. DNS, print) to limit the deployment costs. We are considering all the devices registered with the same co-location server as potentially co-located. Each co-location server is responsible for handling a particular domain, but it is possible that these will overlap or that an impostor might run a server which fails to provide matches as a denial of service attack (we return to security later). These problems can be overcome by performing a search in parallel on all available servers, prioritizing those that provide successful matches in future. A combination of fine-grained deployment of servers, located access (through network schemes) and location services are expected to locate the various devices in the system. Typical semantics of these interactions will involve searching for devices “within x meters”, “the nearest”, “the device labeled y”, or “a device in location labeled z” (where the label is provided by the user). None of these mechanisms is fool-proof and require open access to location systems, user input, or scanning location tags in addition to the system described here. The process of co-location will allow users to reject a choice and get the next alternative – and, of course, to verify that the device is the one they believe it to be.

A problem arises when a registered device (either in paired or unpaired state) moves out of the domain of its current co-location server without performing de-registration with the existing co-location server or before the expiry condition of its registration. Un-pairing will be handled by the paired devices maintenance arrangements, as they may move together – so the pairing correctly does not require the co-location server to continue. De-registration is required to avoid clients attempting to pair with devices which are no longer present. Explicit de-registration is hard to ensure. Expiry will also be provided, but requires a traffic overhead / timeliness trade-off. Where multiple co-location servers have a trust relationship new registrations may cause speculative de-registrations in adjacent domains to smooth the hand-over process. Finally, the server may need to offer an alternative match where a device is no longer available.

3.2 Device Discovery

Discovery mechanisms play an essential role in ad hoc communications. Several discovery protocols have been proposed to facilitate dynamic discovery of services/devices. Some well known discovery protocols include Service Location Protocol (SLP), Secure Discovery Service (SDS), Bluetooth Service Discovery Protocol (SDP), Microsoft's Universal Plug and Play (UPnP) and Jini, Sun's Java-based approach. Each has its own design considerations. For example, SLP and UPnP are designed for TCP/IP networks; while SDS and Jini are restricted to Java applications, and SDP supports only Bluetooth device/service discovery. Detailed comparisons of discovery protocols can be found in [16-18]. Here one can argue that our approach resembles Jini. As a matter of fact, security has not been major goal/objective of Jini and it is based on Java; so, it supports the same weak/light security mechanism as Java offers. Further, non-encrypted Remote Method Invocation (Java RMI) is used for all the communication in Jini that makes it susceptible to eavesdropping, and also Jini does not support resource (service) side authentication. Moreover, in Jini when a client-device wants to create association with the resource-device, the object/programming code is downloaded from the Jini Lookup Table, which is used to pair the devices. This mechanism also introduces a security risk in pairing model as one can launch/put malicious code in the Lookup Table. Service discovery protocols are not the focus of this work, so to simplify analysis of the problem, we decided to focus on our requirements independent of existing technology. After we have proved our solution, we shall incorporate functionality back into existing protocols such as SDP or UPnP if appropriate.

For our initial tests, we used XML to describe the registration and discovery messages mechanism in the proposed architecture. It is portable and flexible enough that we can easily incorporate additional features in the discovery process. Figure 2 shows the XML based device description template and its corresponding DTD document.

During discovery, the device component is responsible for sending an XML-based query to the corresponding co-location server in order to find the required device. When the co-location server receives the client's query, it goes through a match-making process to find the possible matching device(s) in the domain. As a result, if the matching process succeeds, the co-location server generates an XML document with the required information in order to send it to the client and resource for subsequent authentication process. If a compatible device[1] doesn't exist, then the co-location server recommends any possible device(s) by relaxing the strict condition of common co-location capabilities and leaves the client to decide whether to create an association using third party support. If, even after relaxing this condition, there isn't any matching device, the co-location server simply sends a "device not found" message to the client.

(a)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE colocation [
 <!ELEMENT Device (SemanticName,SemanticLocation,CommProtocol,
ColocCapability>

<!ELEMENT SemanticName (Name)>
   <!ELEMENT Name             (#PCDATA)>

<!ELEMENT SemanticLocation (Name)>
   <!ELEMENT Name             (#PCDATA)>

<!ELEMENT CommProtocol(Name,Address)>
   <!ELEMENT Name             (#PCDATA)>
   <!ELEMENT Address          (#PCDATA)>

<!ELEMENT ColocCapability(Name,Type)>
   <!ELEMENT Name             (#PCDATA)>
   <!ELEMENT Type             (#PCDATA)>
]>
```

(b)

```
<Device>
    <SemanticName>
          <Name></Name>
     </SemanticName>

     <SemanticLocation>
           <Name></Name>
     </SemanticLocation>

    <CommProtocol>
          <Name> </Name>
           <Address></Address>
     </CommProtocol>

     <ColocCapability>
            <Name></Name>
            <Type></Type>
    </ColocCapability>
</Device>
```

Fig. 2. (a) DTD for device description (b) XML-based device description template

3.3 Authentication

Authentication is an important part of the pairing process, as it becomes the basis of a secure association between the client and target device. If the authentication process/scheme is weak, then the user can not trust (from security point of view) the pairing system as a whole. In this process, devices exploit the common information received from co-location server to mutually agree on a scheme to generate a key and execute the authentication operation. We are considering a symmetric key to create secure encrypted channel between the devices. Currently, devices generate a key from the data acquired from sensors as suggested by the co-location server during

[1] A pair able device that supports some common co-location capabilities as client for proving its physical existence in the same proximity.

discovery process. Sensors and a key generation algorithm for the devices are selected based on the received information from the co-location server.

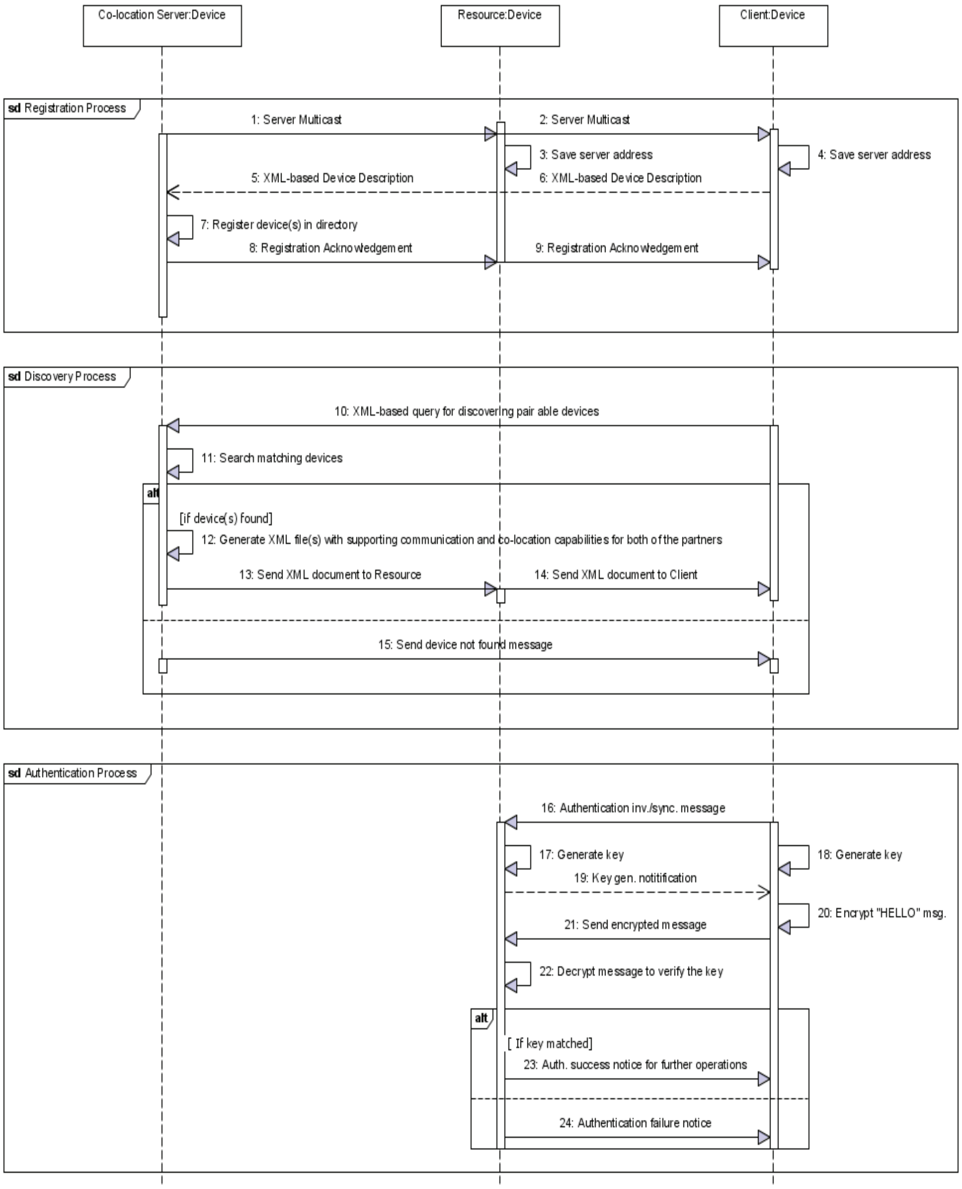
Co-location Server:Device
Resource:Device
Client:Device
sd Registration Process
1: Server Multicast
2: Server Multicast
3: Save server address
4: Save server address
5: XML-based Device Description
6: XML-based Device Description
7: Register device(s) in directory
8: Registration Acknowledgement
9: Registration Acknowledgement
sd Discovery Process
10: XML-based query for discovering pair able devices
11: Search matching devices
alt
[if device(s) found]
12: Generate XML file(s) with supporting communication and co-location capabilities for both of the partners
13: Send XML document to Resource
14: Send XML document to Client
15: Send device not found message
sd Authentication Process
16: Authentication inv./sync. message
17: Generate key
18: Generate key
19: Key gen. notitification
20: Encrypt "HELLO" msg.
21: Send encrypted message
22: Decrypt message to verify the key
alt
[If key matched]
23: Auth. success notice for further operations
24: Authentication failure notice

Fig. 3. Message sequence chart describing the communication pattern for the proposed system

The client-side device component establishes a connection with the intended resource using the communication channel, as described in the received XML from co-location server. Once connection acknowledgement from the resource is received, it sends an authentication invocation message to the resource in order for synchronization and key generation operations to commence. Sensors with same or equivalent capabilities (as recommended by co-location server) on both devices acquire the data from local environment. An encryption key is derived from the collected data samples. When the client receives the key generation completion acknowledgement from the resource, it encrypts a "HELLO" message and transmits it to the resource. The resource decrypts the received message. If the decrypted message is recognized by the resource, the client is authenticated and both the devices enter into paired state. Devising encryption algorithms and generating keys from sensor data is not the focus of our research, so we shall not discuss this further here.

3.4 Security Analysis

Like the schemes for device pairing we build on, we make an assumptions that physical presence and visible actions meet the real access control requirements of the kind of ad-hoc situations described. The devices involved can make use of common sensing capabilities to generate acceptable, strong keys without exposure to third parties or administrators' intervention.

Prior work for device pairing varies greatly in the assumptions about device capabilities, user competence and involvement, as well as security considerations. Understanding the details of various attacks/vulnerabilities in wireless communication is very important in order to determine an appropriate defence strategy for the pairing process. The most significant risk in short range wireless communication (e.g. 802.11, Bluetooth, etc) is that the underlying communication channels are open to everyone including bona-fide users as well as intruders, and thus these cannot be physically secured the same way as a wired network. For example, 802.11 standard uses an encryption system called Wired Equivalent Privacy (WEP). WEP has known vulnerabilities [19], such as it is susceptible to attacks on data and as well as user authentication. These weaknesses allow an intruder to both inappropriately intercept data and also gain access to a network by impersonating a legitimate user. In the case of Bluetooth, devices operate on the 2.4 GHz ISM band. Each Bluetooth device has a unique address, which gives some trust/confidence to user in the identity of the device during association process. For Bluetooth devices to securely associate, an initialization process uses a PIN based approach. Although, the Bluetooth security architecture is relatively secure, it has been vulnerable to key spoofing, address spoofing and PIN cracking [13, 20]. Other threats for wireless communication include well known Man-in-the-Middle (MiTM) and Denial-of-Service (DoS) attacks.

The main goal of an adversary attacking an association model is to fool the legitimate device to associate with adversary's device. Since we are proposing a system for secure device association in close proximity, the threat model considers co-location as the main property to establish a secure channel between two devices. We define the model as follows: two devices that are registered with the same co-location server need to form a secure association between them. By "secure association", we mean that no eavesdropper may decrypt or falsify messages between the communicating partners. We also address the issue of authenticity, which requires that both devices should be able to demonstrate (confirm) the co-location property of each other by the human participants identifying the physical devices involved.

We assume the presence of adversary trying to attack from the same physical space, the next room, the next floor of the building, or possibly from a remote location. Further, it has surveyed the location where the two legitimate devices are attempting to pair and also knows the co-location capability information of the communicating partners. The adversary can use this knowledge to convince one or both of the legitimate devices that it is co-located with them. Since, the problem is demonstrating that two legitimate devices are physically in the same place, verifying that a communicating partner is not an imposter is very important. We consider an impostor attack where the adversary succeeds in pairing with one of the legitimate device by proving falsely that it is physically co-located with it.

Another threat is when a fake co-location server is introduced. This highlight the risk of two possible attacks: denial-of-server (DoS) attack and potential for impersonation attack. We are not considering DoS attack that is result of frequency jamming, since this would affect any communications system. In our proposed solution co-location server only recommends/suggests the common possible method(s) of authentication, but cannot impose any particular scheme. Also, it is not providing any code or information regarding keys to the co-location server, so controlling this device does not provide any privileged information. One possible attack is that a malicious co-location server would only suggest pairing with compromised devices or using weak protocols. Compromised devices are a risk in any system; exclusion of obvious physical devices would cause the server to be questioned; once some basic association has been formed devices may improve the strength of their pairing through maintenance of the connection, which does not require the co-location server. Another possible consideration to mitigate this risk is that each device before registration authenticates the co-location server to check that it is the actual server with which they want to register.

4 Development Status

We have implemented a proof of concept version of the proposed system, which has given us positive results. During these tests, we used PhidgetInterfaceKits along with several sensors and three laptops. Since, the work is still in progress, so more detailed implementation of the system and results has been left for future work.

We want to further clarify that in our proposed scheme, the co-location server only provides bootstrapping information to two unknown devices in an ambient environment, so that pairing process can be commenced. It is the responsibility of device component to execute the authentication scheme to prove the physical co-location property of devices. Moreover, we are not considering the traditional centralized server-based approach. Our proposed system can be implemented with or without directory service. When deployed without a co-location server, peer devices (i.e. client and resource) can locate each other directly using local broadcast or multicast techniques.

Currently, we are investigating a number of authentication strategies to aid the design of our system. Further, we need to consider a number of issues along the way, such as looking into efficient credential revocation mechanisms and device-chaining (i.e. when two devices are in the same proximity but are unable to perform direct authentication because of long distance, then there is the need of another device sharing the proximity with both of the devices to mediate the authentication between them). We are also interested in descriptions of authentication quality (strength of keys, ease of mimicking pairing action, visibility of pairing actions) and their use in selecting mutually acceptable authentication scheme. To aid in the process of determining if the proposed system is successful, we shall use several scenarios that highlight a number of aspects of secure device pairing. We shall also conduct a usability and more detailed security analysis. Results obtained from these analyses will be compared with other existing systems offering pairing mechanism.

5 Conclusion

Pervasive computing has given the vision of *'anytime anywhere'* computing systems, which differ from more traditional computing systems due to the ad-hoc, spontaneous nature of interactions among devices. These systems are prone to security risks, such as eavesdropping but require different techniques to traditional access control to manage. Physical proximity is however a good basis for establishing associations. Many devices will carry sensors for other purposes, which could be used in order to demonstrate this proximity. Recently, secure device pairing has gained significant attention from researchers and a significant set of techniques and protocols have been proposed. Some of these techniques consider devices equipped with infrared or laser transceivers, other require embedded accelerometers, cameras, speakers, microphones and displays. The issue of a universal pairing mechanism is still unresolved. To this end, we attempt to fill the gap left by prior work and propose a general device pairing scheme for pervasive environments. The benefit of this approach from the user's point of view is to eliminate confusion as to what process to follow while pairing devices, and from application and technological point of view is its capability to securely pair the devices under a number of different contexts (in terms of device capabilities).

Acknowledgments. Thanks to Jon Robinson of Software Systems Group for his feedback on this work.

References

1. Goodrich, M.T., et al., Loud and Clear: Human-Verifiable Authentication Based on Audio. in 26th IEEE Intl. Conf. on Distributed Computing Systems, ICDCS 2006.
2. Balfanz, D., et al., Talking to Strangers: Authentication in Ad-hoc Wireless Networks. in Symposium on Network and Distributed Systems Security (NDSS '02). 2002. San Diego, California.
3. Varshavsky, A., et al., Amigo: Proximity-Based Authentication of Mobile Devices. in UbiComp 2007: Ubiquitous Computing. 2007. p. 253-270.
4. Spahic, A., et al., Pre-Authentication using Infrared. in Privacy, Security, and Trust Within the Context of Pervasive Computing, 2005. p. 105-112
5. McCune, J.M., et al., Seeing-is-Believing: Using Camera Phones for Human-Verifiable Authentication. in IEEE Symposium on Security and Privacy, 2005. p. 110 - 124.
6. Mayrhofer, R. and M. Welch. A Human-Verifiable Authentication Protocol Using Visible Laser Light. in 2nd Intl. Conf. on Availability, Reliability and Security (ARES'07) 2007.
7. Saxena, N., et al., Secure Device Pairing based on a Visual Channel. IEEE Symposium on Security and Privacy 2006. Oaklan, CA. p. 306-313.

8. Mayrhofer, R. and H. Gellersen, Shake Well Before Use: Authentication Based on Accelerometer Data. in 5th International Conference on Pervasive Computing (Pervasive-07). 2007.
9. Stajano, F. and R. Anderson, The Resurrecting Duckling: security issues for ubiquitous computing. Computer, 2002. 35(4): p. 22-26.
10. Stajano, F., The Resurrecting Duckling - What Next?, in Revised Papers from the 8th International Workshop on Security Protocols. 2001, Springer-Verlag.
11. Mayrhofer, R., et al., An Authentication Protocol Using Ultrasonic Ranging. Technical Report. 2006, Lancaster University.
12. Mayrhofer, R. and H. Gellersen. On the Security of Ultrasound as Out-of-band Channel. in IEEE International Symposium on Parallel and Distributed Processing (IPDPS-07), 2007.
13. Shaked, Y. and A. Wool. Cracking the Bluetooth PIN. in 3rd ACM Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys '05), 2005. Seattle, Washington.
14. Jakobsson, M. and S. Wetzel, Security Weaknesses in Bluetooth. Lecture Notes in Computer Science, 2001. 2020: p. 176+.
15. Holmquist, L.E., et al., Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts, in 3rd international conference on Ubiquitous Computing. 2001, Springer-Verlag: Atlanta, Georgia, USA.
16. Zhu, F., et al., Classification of Service Discovery in Pervasive Computing Environments. MSU-CSE-02-24, Michigan State University, East Lansing, 2002.
17. Bettstetter, C. and C. Renner. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol. in Proceedings of EUNICE 2000, Sixth EUNICE Open European Summer School. 2000. Twente, Netherlands.
18. Ververidis, C.N. and G.C. Polyzos, Service discovery for mobile Ad Hoc networks: a survey of issues and techniques. in IEEE Communications Surveys & Tutorials, 2008. 10(3): p. 30-45.
19. Borisov, N., et al., Intercepting mobile communications: the insecurity of 802.11. in 7th ACM Annual International Conference on Mobile Computing and Networking (MobiCom '01), 2001. Rome, Italy: ACM.
20. Hager, C.T. and S.F. Midkiff, An analysis of Bluetooth security vulnerabilities. in IEEE Wireless Communications and Networking (WCNC 03), 2003. 3: p. 1825-1831.