A GENERIC FRAMEWORK FOR DEVICE PAIRING IN UBIQUITOUS COMPUTING ENVIRONMENTS

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

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

ABSTRACT

Recently secure device pairing has had significant attention from a wide community of academic as well as industrial researchers and a plethora of schemes and protocols have been proposed, which use various forms of out-of-band exchange to form an association between two unassociated devices. These protocols and schemes have different strengths and weaknesses – often in hardware requirements, strength against various attacks or usability in particular scenarios. From ordinary user's point of view, the problem then becomes which to choose or which is the best possible scheme in a particular scenario. We advocate that in a world of modern heterogeneous devices and requirements, there is a need for mechanisms that allow automated selection of the best protocols without requiring the user to have an in-depth knowledge of the minutiae of the underlying technologies. Towards this, the main argument forming the basis of this research work is that the integration of a discovery mechanism and several pairing schemes into a single system is more efficient from a usability point of view as well as security point of view in terms of dynamic choice of pairing schemes. In pursuit of this, we have proposed a generic system for secure device pairing by demonstration of physical proximity. The contributions presented in this paper include the design and prototype implementation of the proposed framework along with a novel Co-Location protocol.

KEYWORDS

Device Association, Security, Authentication, Physical Proximity, Device Discovery

1. INTRODUCTION AND MOTIVATION

In ubiquitous computing, computing devices are spread around us, whereby they are interconnected with each other through either wireless or wired connectivity. They do not require continuous attention from the users in order to perform tasks as they are seamlessly integrated into the background. Ubiquitous computing environments are becoming popular and a common-place nowadays. It is due to the continuous advancements in communication technologies and proliferation of modern small hand-held devices. Many modern devices (e.g. smart printers, PDAs, smart phones and cameras) support multiple communication channels and almost all of them use wireless technology in some form, such as Bluetooth, Infrared, Wibree, Zigbee, or 802.11. Consequently, over the last ten years significant research efforts have addressed the issue of secure device pairing. The main goal of the research community working on the secure device pairing issue is to provide mechanisms that give assurance of the identity of the devices participating in the pairing process and to secure them from being victims of eavesdropping attacks, such as MiTM attack. Achieving this goal is a challenging problem from both the security and the usability points of view [43, 44].

As a result of these challenges, a wide community of researchers has proposed many protocols and schemes [2-31] to deal with this issue. These protocols vary in their assumptions about the required capabilities in the devices, required human intervention, and in the way they utilize out-of-band or location-limited side channels including physical, audio, visual, short-range wireless channels like Near Field Communications (NFC), and also combinations of these.

Consequently, there currently exists many options for an ordinary user to establish a secure channel between the devices from entering pins and passwords to verifying hashes of public keys and pressing buttons simultaneously on the two devices. This notion contradicts with the usability goal of secure device pairing schemes. As a motivating example towards this, consider the following scenario, which is reproduced from [43].

Let us introduce Angela, who is working in a reputable 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 the 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 ubiquitous 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 process can be applied in each situation. To the best of our knowledge, there is no common secure pairing system that best fits in all four situations of the scenario. For example accelerometer based techniques (e.g. [13, 15, 18]) are not practical for large devices, and in a large room with a roof mounted projector radio signal and close-range techniques are likely to fail (e.g. [14, 28]). Where a choice of pairing techniques is available not all users are capable 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.

In view of above facts, we believe that a common pairing infrastructure for ubiquitous computing environments can improve the usability of the pairing process. In this piece of work we are presenting such a system. The proposed system integrates device discovery, several pairing schemes and a selection mechanism into a single model that facilitates association of any pair of devices in a wide range of scenarios by using the devices' existing capabilities and user preferences, and also assists the user to select an appropriate pairing protocols and relieves him/her from choosing between more than two dozen [2-31] of pairing schemes. The interested readers can find the detailed analysis of these existing schemes in [43, 44].

2. THE NEED FOR A FRAMEWORK BASED APPROACH TO DEVICE PAIRING

Each of the proposed schemes we have surveyed [43, 44] has strengths and weaknesses – often in hardware requirements, strength against various attacks or usability in particular scenarios. Therefore, we can conclude that no one has yet devised a pairing protocol, which is generic enough to accommodate a very large set of device pairing scenarios and can be considered as a standard solution for ubiquitous computing environments. Currently available schemes for secure device pairing 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 as well as the required proximity between the devices. Some of these techniques consider devices equipped with infrared, laser or ultrasound transceivers, whilst others require embedded accelerometers, cameras and/or LEDs, display, microphone and/or speakers. Some techniques exploit the knowledge of radio environment to securely pair the devices; others require the user's careful attention and significant manual intervention in pairing process. Further, most of the prior work on secure device pairing considered demonstrative approach (i.e. requires user involvement and/or manual efforts to identify the intended partner) to identification and discovery of the intended pair able co-located device. For example in SiB [29] and the Resurrecting Duckling Security Model [3], the discovery of the intended pair able device is performed manually; while in Talking to Strangers [21] communicating partners exchange their connectivity information over the secondary channel (i.e. infrared). However, in many situations automatic device discovery is required [7]. If we continue to multiply the number of manual or out-of-band discovery mechanisms, users will become confused about the selection of device discovery method during pairing process. For instance, a user wanting to create an association of a mobile phone having a microphone, speaker, camera, display and infrared with another mobile phone having microphone, speaker, display, no camera and no infrared might be confused about the varied types of manual or out-of-band possibilities for device discovery [7]. We therefore agree with the view proposed by Saxena et al. [7] that it should not be the user's responsibility to figure out how and which method to use for device discovery each time; instead an automatic device discovery should take place.

It is therefore appropriate to investigate ways of integrating different pairing protocols and discovery mechanism within a general architecture for providing secure and usable pairing mechanisms for a large set of ad hoc scenarios in ubiquitous computing environments. Such architecture should facilitate choice of the best pairing scheme, considering device capabilities, environmental limitations, user preferences and the balance between security and usability. We realized this need and proposed a framework-based approach to deal with this issue.

3. SYSTEM DESIGN AND ANALYSIS

3.1 Design Goals

The major goal of this research is to design a system that facilitates association of any two co-located devices by demonstration of physical proximity through the integration of discovery mechanism and Proof-of-Proximity (PoP) schemes. Note that PoP schemes are either derived/extended from existing pairing protocols or taken in their original form to provide the authenticity of the physical proximity of devices. These pairing schemes exploit various forms of Out-of-Band (OOB) channels. In the literature of device pairing OOB channel refers to a secondary channel, that work along with the primary in-band channel, such as InfraRed or Bluetooth, with additional security guarantees. Due to these features, OOB channels are helpful in developing secure device pairing protocols/schemes.

The three main goals of the proposed system are described below:

- **Generality:** Generality is one of the main goals of the proposed system. The system should be applicable in a wide range of device pairing scenarios in ubiquitous computing environments, capable of incorporating existing pairing schemes and can be extended without major modifications in the design.
- **Usability:** From a usability point of view, the system should be simple to understand, and easy to use for an ordinary user.

- **Security:** Our security goal is twofold. Firstly, the system should be capable of establishing the secure session between two previously unassociated devices through proving the physical proximity of the devices involved in the pairing process. Secondly, all the communication between the entities of the system must be secured.

3.2 Design Requirements

To achieve the above mentioned goals, we have identified some of the major requirements described below:

(A) A mechanism is required that facilitates the discovery of possibly co-located devices in the vicinity.

To meet this requirement, we have designed a simple registration and discovery mechanism, which is presented in section 3.5.4.

(B) A set of protocols or schemes is required that demonstrates the physical proximity of the two devices.

We have already published a detailed survey of the state-of-the-art in device pairing [43, 44], in which a detailed list of pairing protocols is presented. These pairing protocols and/or their variations could be used to demonstrate the physical proximity of devices. Hereafter in this paper, the term 'PoP protocols' refers to the set of pairing protocols that are implemented in the proposed system either in their original form or with some variations to facilitate the demonstration of physical proximity through the use of OOB channels.

(C) A generic protocol is required that integrates the discovery mechanism and a set of PoP protocols and exchanges all the other required information/messages between several entities of the system in an encrypted form.

We have designed the Co-Location (CoLoc) protocol to meet this requirement, which is presented in section 3.6.

(D) Users should have some control on the selection of PoP protocols, and the level of required user interaction.

The proposed system is capable of getting user's preferences and considers them during the PoP protocols selection phase. We have presented the details of the selection mechanism of PoP protocols in sections 3.6.5 and 3.7.

(E) The ability to modify PoP protocol selection behaviour at run-time.

The protocol selection mechanism uses an XML-based policy as PoP protocols selection criteria, which is defined in terms of required device capabilities and constraints over PoP protocols. Since, the criterion for the selection of PoP protocols is described in an XML-based protocol specification and selection policy file; it can be changed / modified at run-time.

3.3 Design Assumptions

We are considering ubiquitous computing environments, in which devices communicate with each other through short-range wireless technology, such as 802.11 or Bluetooth. They discover each other using our proposed registration and discovery mechanism. We are not considering extremely resource constrained devices, such as sensor nodes. Instead, we are considering those ubiquitous computing devices, which have reasonable battery power and computational capabilities, e.g. mobile phones, cameras, PDAs, laptops, printers etc. These devices are capable of symmetric encryption/decryption, public key based encryption, hashing, signature verification, and have unique device-id or address. Further, devices know their location through some location system already installed in the environment or through their own hardware/software, such as GPS (Global Positioning System). The location information is useful in the discovery process. We assume that the co-location server is a trusted, uncompromised and tamper resistant (or at least tamper evident) device. It is also capable of performing symmetric and asymmetric cryptographic operations. Since, the co-location server is very light-weight; it might be run with other local services (e.g. DNS, print) or any other server, which is part of some existing security infrastructure to limit the deployment costs. Alternatively, it could also be installed into a dedicated low-cost small device. Then each device needs to perform one time demonstrative discovery of the server device in order to build trust. We are considering all the devices registered with the same co-location server as potentially co-located and each co-location server is responsible for handling a particular domain or location. We believe that due to the modern low-cost small ubiquitous computing devices that have now reasonable battery and computational power, one co-location server per scope is feasible.

3.4 High-level System Design

In figure 1 (left), we have shown the high-level architecture of the proposed system, which illustrates three phases. The first two phases are registration and discovery of the device(s), and the third phase is selection, initiation and execution of the PoP protocol. Figure 1 (right) shows the two major components of the system, which are the co-location server and the device. These two components are composed of several other software components.

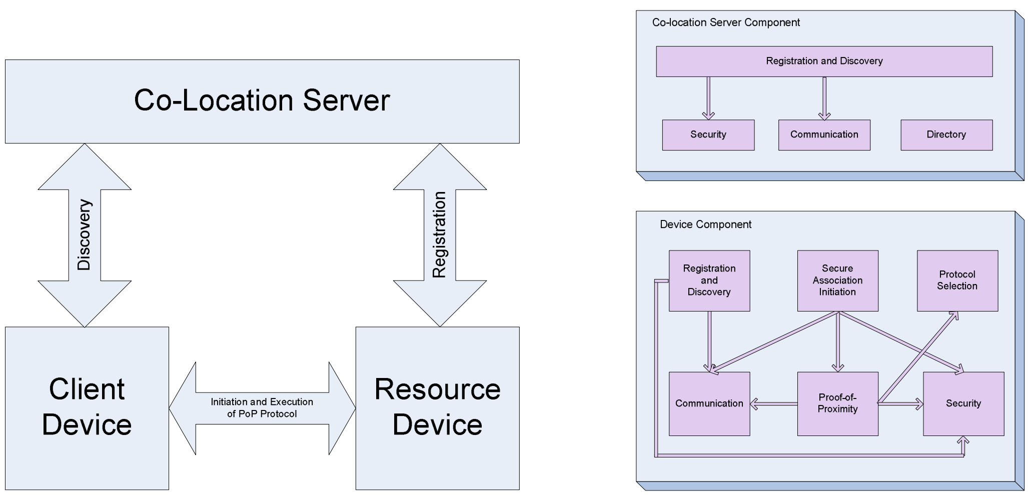

Figure 1: High-level architecture (left) and two major components (right) of the proposed system

The registration, discovery and proof of physical proximity are integrated into Co-Location (CoLoc) protocol, which is core part of the proposed system and one of our main contributions as well. The CoLoc protocol is described in section 3.6 in detail, however we are presenting the overview of the overall system as below:

1. First of all resource device(s) register their capabilities with an easily found database stored on the co-location server. New devices can be added while the system is running.
2. When two devices need to associate, the client queries the co-location server to acquire the required information of suitable resource device(s).
3. The co-location server prepares a device list containing necessary information for selecting and contacting the resource device in order to initiate the proof-of-proximity phase.
4. Based on the information from the co-location server and user preferences, the client first goes through the PoP protocol selection process and then initiates the secure association

initiation process with the selected resource device. Different interactions to demonstrate physical proximity are possible and the selection requires a selection criterion along with device capabilities, constraints on pairing schemes and/or user preferences.

5. Both of the devices (i.e. client and resource) execute the commonly agreed PoP protocol for the purpose of demonstrating their physical proximity in order to establish the secure session. Note that secure pairing is achieved only when physical proximity between both of the devices is proved.

3.5 Device(s) Registration and Discovery Mechanism

There is an immense literature on service discovery in general; however during the last ten years many discovery protocols have also been proposed to facilitate dynamic discovery of services and/or devices in ubiquitous computing environments. Some well known discovery protocols include Service Location Protocol (SLP) [32], Bluetooth Service Discovery Protocol (SDP) [33, 34], Microsoft's Universal Plug and Play (UPnP) [35] and Jini [45, 36]. Each has its own design considerations. For example, SDP supports only Bluetooth device/service discovery; while Jini is restricted to Java applications, SLP and UPnP are designed for TCP/IP networks; however UPnP is targeted to small or home based computing environments, while SLP is targeted to both from small to large-scale enterprise networks. Detailed comparisons of discovery protocols can be found in [37-39]. However for the sake of completeness and motivation, we are presenting some of the relevant discovery protocols followed by the proposed device registration and discovery mechanism.

3.5.1 Service Location Protocol (SLP)

Service Location Protocol (SLP) [32, 38, 40] is a discovery protocol developed by the IETF (Internet Engineering Task Force) working group for service registration and discovery within a particular location or scope. It is designed for small to large scale enterprise networks. There are three major components of SLP, which are known as agents: Directory Agent (DA), User Agent (UA), and Service Agent (SA). DA is responsible for providing the directory services. SA advertises the location information along with the service-attributes on behalf of a service through registration process, and UA on behalf of the client application sends service discovery requests to a DA.

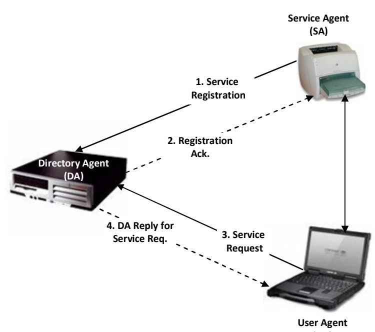

Figure 2: An illustration of service registration and discovery mechanism in SLP

Figure 2 illustrates the interaction mechanism between the three components of SLP in a small or Local Area Network. First of all DA announces its presence through periodic scoped-multicasting on a well known channel. A UA or SA discovers the address of the DA through some mechanism, such as listening to a DA advertisement message passively or actively multicasting discovery message to the SLP multicast network address (i.e. 239.255.255.253). It is also possible to configure the DA address statically through Dynamic Host Configuration Protocol (DHCP) [41].

Once an SA discovers a DA, it registers with it by sending a service registration message. Service advertisements are made through the use of a service URL and service template. The registration message contains the URL for the advertised service including its lifetime. An SLP service is described in the form of set of service attribute-value pair. A sample SLP service template for a print service is given below:

```
service:printer://lj2420dn.FONT.susx.ac.uk:1024/
scopes = FONT, administrator
printer-name=lj2420dn
printer-network-name = Inf-pev-5c4-bw
printer-location = Pevensey II, Room 5C4
color-supported = true
...
...
...
```

As stated, when a UA requests a service, it contacts a known DA by sending a service request/query to obtain the service URL. Once the UA receives the service URL, it can access the service pointed to by the returned URL. DA is an optional component in SLP; therefore, in the scenarios where there is no DA available, the UA and SA discover each other directly through a multicast mechanism.

3.5.2 Jini Technology

Jini [36] is a java-based service registration and discovery technology developed by Sun Microsystems. Jini provides service/device registration, discovery and communication mechanisms for ad hoc networks. The core part of Jini technology is a set of protocols known as discovery-join-lookup.

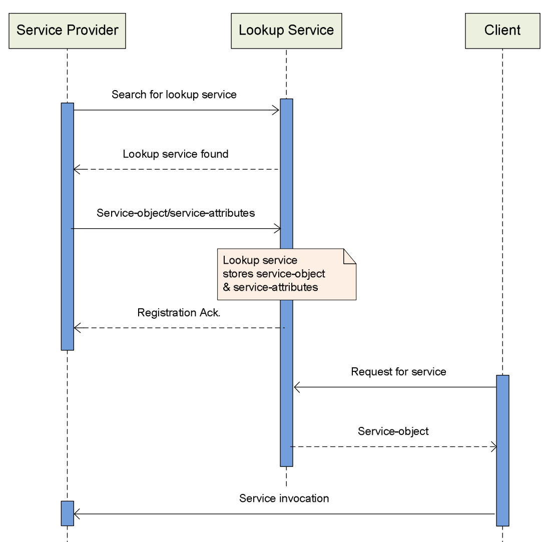

Figure 3: Message sequence diagram illustrating Jini's discovery-join-lookup process

Figure 3 illustrates the functionality of these protocols. On bootstrapping, services look for a lookup service and register themselves with it. This process is known as the Discovery and Join process. During the registration process Jini services upload their service-object along with service attributes in a Lookup Table of the lookup service. Then, when a client needs a service, it also looks for a lookup service to find out the required services and to download the service-object. Once the client downloads the service-object from the lookup service, it directly contacts the service for further communication. This is known as the Lookup process. As in SLP, Jini lookup servers containing Lookup Tables serve the purpose of a directory. However, unlike SLP, Jini does not support directory-less mode and it always needs at least one lookup service. Further, in contrast to SLP, Jini services are described in Java.

3.5.3 Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP) is a device-centric peer-to-peer technology developed by the UPnP Forum [35]. Microsoft Corporation played an important role in UPnP's development and it is considered as an extension to the Microsoft's Plug and Play technology; however it is more than just an extension. The major objective of UPnP is to enable discovery, auto-configuration, management and control of devices in unmanaged and small computing environments, such as small office or home environments. UPnP achieves its goal through utilizing existing standards, such as web and TCP/IP technologies. For service/device discovery, it uses Simple Service Discovery Protocol (SSDP) [42]. As in Jini's discovery-join-lookup process, SSDP is used for both advertising the device's (service) presences to the other devices in the proximity/scope as well as discovering other peer devices. However, unlike SLP and Jini, UPnP does not require any central repository to store the service or device information and/or service-object. Further, in contrast to SLP and Jini, UPnP uses XML for all the communication and exchange of device's information among the two entities of UPnP network (i.e. Control Point and Device). Devices' profiles describing their capabilities and features are written in XML format. Interested readers can refer to [35] for details of the UPnP device architecture provided by UPnP forum.

3.5.4 The Proposed Device Registration and Discovery Mechanism

When we analyze the previously described discovery schemes, it is noted that security has never been a major concern or major design goal of these technologies. For example, Jini uses non-encrypted Remote Method Invocation (Java RMI) for all the communication that makes it susceptible to eavesdropping. Additionally, when a client wants to create an association with the resource/service, as a part of this process the service-object is downloaded from the Jini lookup service and this introduces the overhead in the sense that small devices have scare resources, and also there is a security risk in that an adversary can register a bogus service containing malicious code as its service-object. Further, it is also noted that at very basic level, the architecture of these discovery protocols is similar; however each has some of its own assumptions and features that make it feasible for implementing in particular scenarios or environments. For example, SLP and UPnP are targeted to IP based device environments, while Jini is not restricted to IP-based environments; however it requires JVM (Java Virtual Machine). UPnP utilizes XML technology for device/service registration and discovery, while it is not the case in SLP and Jini. In Jini, the client requires more processing capability as compared to UPnP and SLP due to the installed JVM and downloaded service-object. It is not a big deal for large devices, such as desktop computers or laptops; however it is still challenging for small resource-constrained devices.

In summary, to simplify the analysis, design and prototype implementation of the proposed framework to test our hypothesis, we decided to design our own registration and discovery mechanism. Our proposed discovery and registration system incorporates several similar features to the device discovery technologies discussed above along with some of its own unique features to make the registration and discovery process simple, easy to implement, independent of existing technologies, and confidentiality and integrity protected. For example, like UPnP we have used XML to describe the registration and discovery messages mechanism for the proposed system, however in a much simpler way than UPnP. The reason we use XML is that it is an advantage for any modern communication system due to its flexibility, programming language independence and portability characteristics. It is flexible enough that one can incorporate additional features in the system later on, if necessary, and it also significantly increases interoperability between systems.

Further, unlike Jini, where devices look for a lookup service through multicasting a search request on the network, in the proposed system the co-location server advertises itself through multicasting. We choose this mechanism as transmitting data consumes more battery power as compared to receiving data, and the devices in ubiquitous computing environments are more battery-constrained as compared to the server or base station. In our proposed system, the registration process could be considered equivalent to Jini's discovery and join process, and the discovery process could be considered equivalent to Jini's lookup process. Also note that in our proposed mechanism, all the communication during registration and discovery process between several entities of the system is encrypted, which is described in next section.

3.6 Co-location (CoLoc) Protocol

The co-location (CoLoc) protocol is a core part of our system and one of our main contributions. It is designed to achieve our generality, usability and security goals. It provides the functionality of registration, discovery, and security association initiation and execution of the selected PoP protocol. For the sake of simplicity and clarity, we have divided the overall protocol into three parts: registration, discovery of intended pairable device, and the selection and execution of an appropriate protocol to demonstrate/authenticate the physical proximity. The selection process involves device capabilities, constraints on pairing schemes and/or user preferences. The detailed description of each of the parts can be found in subsequent sections preceded by the description of several notations used in describing the CoLoc protocol.

3.6.1 Notations

CS: Co-location server
A: Resource device
B: Client device
Process_i: Actions/processes performed at device *i* before sending or receiving a message.
X → Y: Msg : A message Msg sent from X to Y over a communication channel.
PK_i: Public key of *i*.
K_i: Private or secret key of *i*.
SK_i: Session key internally generated by *i*.
PSK: Pairing session key.
PSK_{ij}: Shared pairing session key for the parties *i* and *j*.
CP: Credential password, shared among all the registered devices and co-location server.
Enc(): Encryption function.
Dec(): Decryption function.
Enc(x)y: An encryption function that encrypts plaintext x using key y, which could be a public/private key or shared secret key.
Dec(x)y: A decryption function that decrypts ciphertext x using key y, which could be a public/private key or shared secret key.
MAC(x)y: A keyed message authentication function that is applied to x using key y.
||: Concatenation operator

3.6.2 Bootstrapping

Bootstrapping in our system refers to the initialization and advertisement of the co-location server. During bootstrapping, the co-location server generates its public/private key pair (i.e. PK_{Coloc} and K_{Coloc}) and broadcasts its connectivity information along with its public key. Then, devices discover the co-location server for registration and/or discovery tasks by listening to the broadcast messages. Alternatively, in certain scenarios, where this mechanism is not available or difficult to implement, a one-time demonstrative discovery of the co-location server can be performed, which has now become more common in the literature of device pairing. In this approach, the user is involved in identifying and obtaining the connectivity information of the resource or intended communicating partner through some manual effort.

3.6.3 Registration and Discovery Part of the CoLoc Protocol

Figure 4 shows the registration part of the CoLoc protocol. Once the system is bootstrapped and device A receives the public key PK_{Coloc} of the co-location server, it encrypts the device profile with an internally generated temporary session key SK_A. Then it sends an encrypted message along with a message authentication code (MAC), and SK_A encrypted with the co-location server's public key PK_{Coloc} to the server. The device profile contains the id of device A along with connectivity information and some keywords (user friendly names) to identify the device in the networked environment, capability information (such as camera, display, keypad, etc), lease duration and optionally device location information (such as Pevensey II, Room 5c11, etc). Additionally, any constraints or user input/preferences are also injected in the DeviceProfile.

Process_CS:
Generate public/private key pair: PK_{Coloc}, K_{Coloc}
CS → A: ServerID,PK_{Coloc}
Process_A:
RegistryMsg:=Enc($DeviceProfile_A$)SK_A|| Enc(SK_A)PK_{Coloc},
MAC(RegistryMsg)SK_A
A → CS: RegistryMsg
Process_CS:
splits and Decrypt *RegistryMsg* to obtain SK_A first, and then *DeviceProfile.*
If(integrityChecked()), then
(a) Generate: PSK_A
(b) Register Device
(c) ResponseMsg:= Success || PSK_A || CP
Else
(a) ResponseMsg:= Abort/Fail
End If
RegAck:= Enc(ResponseMsg)SK_A
Destroy session key: SK_A
CS → A: RegAck
Process_A:
Dec(RegAck)SK_A to obtain PSK_A

Figure 4: Registration part of the CoLoc protocol

The co-location server splits and decrypts the registration message in order to obtain the SK_A first, which is then used to obtain the device profile. The co-location server also performs integrity check before registering the device. In response to a registration request, the co-location server sends an acknowledgement message to the device A, containing a one-time pairing session key PSK_A and credential password CP encrypted with temporary session key SK_A. Credential password CP is used in revocation mechanism. The registration process applies to every device intended to become part of the deployed ubiquitous system. Once registration is done, device A will be visible to the other devices through the querying co-location server.

In the discovery process (as shown in figure 5), device B (client) encrypts a query with temporary session key SK_B. Then, it encrypts SK_B with PK_{Coloc} and sends it to the co-location server along with the encrypted query and MAC of the overall message. The co-location server decrypts the client message and also performs an integrity check before going through the match-making process based on the criteria given in the query. Query contains the user-friendly name (if known) or the type of device, any user preferences for pairing process and optionally the locations in which devices should be searched (if server domain is too broad). As a consequence, the co-location server sends information on matching devices (referred to as a

ResultSet hereafter in this paper) to the client (i.e. device B) encrypted with SK_B. ResultSet contains the profiles of found devices based on the criteria given in query along with their one-time pairing session keys (i.e. PSK_i) and their expiry time.

Process_B:
DiscoveryReq:=Enc(Query)SK_B || Enc(SK_B)PK_{Coloc},
MAC(DiscoveryReq)SK_B
B → CS: DiscoveryReq
Process_CS:
Split and Decrypt *DiscoveryReq* to obtain SK_B first, and then *Query*.
If(integrityChecked()), then
(a) Generate *ResultSet* containing matching devices' profiles and their corresponding pairing session keys along with expiry time.
(b) DiscoveryRespMsg:= Enc(ResultSet)SK_B
Else
(a) DiscoveryRespMsg:=Abort/Fail
End If
Destroy session key: SK_B
CS → B: DiscoveryRespMsg
Process_B:
Dec(DiscoveryRespMsg)SK_B
to obtain devices' information along with their pairing session key(s) and their expiry time.

Figure 5: Discovery part of the CoLoc protocol

3.6.4 Registration Renewal, Update and Device De-Registration

In the proposed system, the registered devices are capable of renewing or updating their registration. Renewal and update requests can be for updating/modifying the device's profile or device status (i.e. busy or available), and extension/renewal of the lease time and/or pairing session key. Explicit de-registration can be performed on the demand of the registered device by sending a de-registration request to the co-location server. The co-location server also performs implicit de-registration when the device lease time expires to keep the registered devices' information up-to-date, and to maintain the device's directory. During the implicit de-registration process, any device whose lease time expires is automatically de-registered by the co-location server by deleting their entry from the directory.

3.6.5 Selection and Execution of Mutually Agreed Scheme

As shown in figure 6, during this phase the client sends a message, containing the name of the selected PoP protocol, to the resource to initiate the pairing process. Once the resource device receives that message, it starts generating PoP data that will be used to verify the physical proximity of the devices. PoP data could be generated in numerous ways based on the nature of agreed protocol. For example, many modern devices carry sensors for other purposes, which could be used to obtain the PoP data. Where sensors are not available or it is hard to obtain PoP data directly from sensors, then user could be involved to get the PoP data. Considering the nature and ways of demonstrating the physical proximity of devices, PoP protocols are classified into four categories. The first category belongs to those protocols, which require user involvement in only generating PoP data, such as Button-to-Button and Blink-to-Button [43, 44, 50]. In that case, verification of PoP data is done internally by the system. The second category belongs to those schemes which require user involvement only in verification of PoP data, such as Display-Display and Blink-Blink [43, 44, 50]. In that case PoP data is generated either internally by the system or from attached sensors with the devices. The third category belongs to those schemes, which require the user to be involved in generating PoP data as well as in verifying that data, such as Capture and Show [50]. The fourth category belongs to those schemes, which do not involve the user in the proof-of-proximity process at all, so we call them automatic pairing schemes. At the end of execution of this phase, if the physical proximity has been proved, the established session between both of the devices is considered to be secure.

Process_B:
Execute an algorithm that seeks for the best possible available mutually supported schemes that could be
used to authenticate/demonstrate physical proximity between the devices.
Inv_PairingMsg:=Enc(SelectedProtocolName)PSK_{AB}

B → A: Inv_PairingMsg
Process_B:
Generate PoP data to verify the physical proximity based
on the nature of agreed PoP protocol.
Process_A:
Dec(Inv_PairingMsg)PSK_{AB} to retrieve the name of
pairing scheme.
Generate PoP data that will be used to verify the
physical proximity between the devices.
Resp_Msg:= ENC(Resp_InvPairingMsg)PSK_{AB}

A → B: Resp_Msg
Process_B:
DEC(Resp_Msg)PSK_{AB}
ProximityVerMsg:= ENC(PoP data)PSK_{AB}

B → A: ProximityVerMsg
Process_A:
DEC(ProximityVerMsg)PSK_{AB}
Perform demonstration of physical proximity
based on the previously agreed PoP protocol.
If(success)
    pairingA = 'Accepted'
Else
    pairingA = 'Rejected'
End If
ProximityVerResp:= ENC(pairingA)PSK_{AB}
Process_B:
Perform demonstration of physical proximity
based on the previously agreed PoP protocol in parallel
to device A.
If(success)
    pairingB = 'Accepted'
Else
    pairingB = 'Rejected'
End If

A → B: ProximityVerResp
Process_B:
DEC(ProximityVerResp)PSK_{AB} to obtain pairingA.
Pairing is achieved/granted only when
    (pairingA= 'Accepted' & pairingB = 'Accepted')
Otherwise B aborts the pairing process

Figure 6: Secure association initiation and execution of PoP protocol

3.7 Selection of PoP Protocol(s)

As described earlier, once the device discovery operation completes, the subsequent phase is the selection and execution of the PoP protocol. To achieve the objective of selecting an appropriate PoP protocol, we have designed a protocol selection algorithm presented in figure 7. The input parameters of algorithm include client's own device profile, resource device profile and an XML-based PoP protocol specification and selection policy.

A sample protocol specification and selection policy is shown in figure 8. <Name> tag contains the name of the PoP protocol for which other tags describe the selection criteria. The value of <Type> tag represents one of the categories of PoP protocols, which are briefly described in previous section and other details of these categories is given in chapter 4. The values of <ClCapabilities> and <ResCapabilities> tags describe the required capabilities of client and resource devices for the execution of the protocol. The value of <ProximityLimit> tag represents the maximum distance between the pairing partners up to which the protocol can work or can achieve good results. The value of <ProximityLimit> is given in centimeters. The value of <UILevel> represents the level of required user interaction. '1' represents the low or minimum level of user interaction and '3' represents the high or maximum level of user interaction. These values are obtained based on the classification of PoP protocols through a usability study, the details of which are out of the scope of this paper.

Input:

- Client's own profile,
- Resource's profile,
- Protocol specification and selection policy

Output:

RecommendedProtocol(s) based on the given input parameters

Step 1: Cl_Profile:= Read client's own profile

Step 2: Res_Profile:= Read resource's profile (i.e. received from Co-location server)

Step 3: Cl_SupportedProtocols:=getProtocolList(Cl_Profile)

Step 4: Res_SupportedProtocols:=getProtocolList(Res_Profile)

Step 5: Comm_SupportedProtocols:=mutualProtocolList(Cl_SupportedProtocols, Res_SupportedProtocols)

Step 6: RecommendedProtocols:=getBestProtocols(Comm_SupportedProtocols, Cl_Constraints/Preference, Res_Constraints/Preferences)

Step 7: Return: RecommendedProtocols

Figure 7: An algorithm to find out the best possible PoP protocol(s) based on given input parameters

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<PSPolicy>
    <Protocol>
        <Name>Button_to_Button</Name>
        <Type>1</Type>
        <ClCapabilities>Button</ClCapabilities>
        <ResCapabilities>Button</ResCapabilities>
        <ProximityLimit>100</ProximityLimit>
        <UILevel>1</UILevel>
     </Protocol>
    <Protocol>
        <Name>Capture_And_Show</Name>
        <Type>3</Type>
        <ClCapabilities>Camera;Display</ClCapabilities>
        <ResCapabilities>Display</ResCapabilities>
        <ProximityLimit>200</ProximityLimit>
        <UILevel>3</UILevel>
    </Protocol>
    ...
    ...
    ...
```

Figure 8: A sample protocol specification and selection policy

3.7.1 Internal Working of Protocol Selection Algorithm

The protocol selection algorithm is consisting of several rounds. Each round facilitates with the filtration process of PoP protocols. During each round those PoP protocols are discarded which does not meet the requirements of some particular constraint of that round. Ultimately in round-

5, we obtain those PoP protocols, which satisfy the user preferences and other requirements of the scenario/situation in which pairing process is going to be occurred. We describe each round of the execution of PoP protocol selection algorithm as below:

Round-1: (Input: client-device profile, resource-device profile, PoP protocols specification and selection policy)

Filter/select PoP protocols based on required capabilities of client and resource devices.

Round-2: (Input: selected PoP protocols from Round-1 and the PoP protocols specification and selection policy).

Select the PoP Protocols that are appropriate for working within the given distance. It is achieved through comparing and performing selection based on the value of <ProximityLimit> tag with the distance given/input by the user.

Round-3: (Input: selected PoP protocols from Round-2 and the PoP protocols specification and selection policy).

Select PoP protocols based on the level of required user interaction during pairing process. It is achieved through comparing the value of UILevel tag with the user-interaction option as selected/given by the user.

Round-4: (Input: selected PoP protocols from Round-3 and PoP protocols specification and selection policy).

Select PoP protocols based on the constraints/limitations of PoP protocols and user preferences. It is achieved through comparing the value of <Constraints> tag with the given user preferences.

Round-5: (Input: selected PoP protocols from Round-4 and PoP protocols specification and selection policy).

In this final round, the priority level/recommended order is assigned to each of the PoP protocols obtained from Round-5. The high-level description of the calculation process for priority-level is described below. Note that the scores/points used in these calculations are only for demonstration and proof-of-concept purposes.

- **PoP protocol points calculation process from security point of view:**

```
If(fatal errors are not applicable to PoP protocol)
        FatalErrorPoints = 4;
Else
        FatalErrorPoints = 2;
        EndIf
If(safe errors are not applicable to PoP protocol)
        SafeErrorPoints = 2;
Else
        SafeErrorPoints = 1;
EndIf
```

Note that the points for fatal errors and safe errors differ from each other due to the fact that fatal errors are more dangerous and serious than safe errors.

- **PoP protocol points calculation process from execution-time point of view:**

```
If (ProtocolExeuctionTime <= 15 seconds)
        ExecutionTimePoints = 5;
ElseIf (ProtocolExeuctionTime > 15 seconds and <= 30 seconds)
        ExecutionTimePoints = 4;
```

```
ElseIf (ProtocolExeuctionTime > 30 seconds and <= 45 seconds)
        ExecutionTimePoints = 3;
ElseIf (ProtocolExeuctionTime > 45 seconds and <= 60 seconds)
        ExecutionTimePoints = 2;
ElseIf (ProtocolExeuctionTime > 60 seconds and <= 75 seconds)
        ExecutionTimePoints = 1;
Else
        ExecutionTimePoints = 0;
EndIf
```

Based on the above mentioned points calculation process, the level of priority is calculated, which eventually sets the recommended order of the PoP protocols. Rule is that the PoP protocol that has highest score/points will be the best protocol for a given scenario/situation.

5. IMPLEMENTATION AND EVALUATION

To evaluate the proposed system, we built a prototype implementation of the system and conducted a usability study. However, the details of the usability study are focus of another paper. The implementation of the proposed system is carried out using Java (version 1.6) and Windows XP operating system. In the coding and implementation process, we have used Eclipse Galileo (version 3.5) as a Java IDE. As an apparatus, we have used two 1.9GHz Dell Machines with 1GB RAM, two PhidgetInterfaceKits [46] and a camera. PhidgetInterfaceKits and camera are the requirement for some of the PoP protocols. At server-side Oracle Berkeley DBXML [47, 48] is used to maintain and keep record of the devices' profiles. The implemented system integrates 14 different pairing schemes to demonstrate the physical proximity of the devices. The software components of the proposed system are implemented as Java packages. However, due to the space limits we cannot describe the details of the 14 implemented PoPprotocols and the functionality of the software components in this paper.

5.1 Structure of the CoLoc Protocol Messages

The DTD to validate the CoLoc protocol messages is shown in figure 9. This DTD describes the structure of CoLoc protocol messages and is used to validate all the XML-based communication in the system during registration, discovery and proof-of-proximity phases, such as device profiles, discovery queries, and co-location server's reply to client in response of discovery request.

```
<!-- DTD validates protocol messages-->
<!ELEMENTProtocolMsg ( Command, XMLData? ) >
<!ELEMENT Command ( #PCDATA ) >
<!ELEMENTXMLData ( DeviceProfile | Query | DeviceList | DeviceID | PoPProtocol | PoPData | PSK | CP) >
<!ELEMENTDeviceProfile ( DeviceID, LDuration, Keyword, DeviceLocation?,CommProtocol+,DeviceCap, UserInput?)>
<!ELEMENT Query ( #PCDATA ) >
<!ELEMENTDeviceList ( DeviceInfo+ ) >
<!ELEMENTDeviceInfo ( PSK, ExpiryTime, DeviceID, DeviceLocation?,CommProtocol+,DeviceCap, UserInput?)>
<!ELEMENTExpiryTime ( #PCDATA ) >
<!ELEMENT PSK ( #PCDATA ) >
<!ELEMENTDeviceID ( #PCDATA ) >
<!ELEMENTLDuration ( #PCDATA ) >
<!ELEMENT Keyword ( #PCDATA ) >
<!ELEMENTDeviceLocation ( #PCDATA ) >
<!ELEMENTCommProtocol (ChannelName, Address) >
<!ELEMENTChannelName ( #PCDATA ) >
<!ELEMENT Address ( #PCDATA ) >
<!ELEMENTDeviceCap ( #PCDATA ) >
<!ELEMENTUserInput ( #PCDATA ) >
<!ELEMENTPoPProtocol ( #PCDATA ) >
<!ELEMENTPoPData ( #PCDATA ) >
<!ELEMENT PSK ( #PCDATA ) >
<!ELEMENT CP ( #PCDATA ) >
```

Figure 9: DTD to validate the messages of CoLoc protocol

Every Coloc protocol message contains a 'Command' tag and at most one 'XMLData' tag. Command tag defines the type of message or protocol instruction, while the XMLData tag defines several other sub-tags, but only one can be used in any given message. For example, XMLData tag contains a 'DeviceProfile' sub-tag to define the device profile during registration, and a 'Query' sub-tag to define the client's query during the discovery phase. 'DeviceList' tag defines the list of found devices as a result of client's query. The DeviceID tag is used when a resource device performs explicit de-registration with the co-location server or request for a renewal of registration or pairing session key. The PSK tag defines the pairing session key and is used during the registration or the renewal of registration of the resource device, while CP tag defines the credential password that is used in providing credential revocation mechanism. PoPProtocol and PoPData are used during the proof of proximity phase, which define PoP protocol name and PoP data respectively. A CoLoc protocol message that illustrates the device's explicit deregistration request is given in figure 10.


```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<ProtocolMsg>
    <Command>DeRegister</Command>
    <XMLData>
        <DeviceID>Wiston</DeviceID>
    </XMLData>
</ProtocolMsg>
```


Figure 10: A Coloc protocol message illustrating the device's explicit deregistration request

5.2 Demonstration of Prototype Implementation

We have designed simple user interfaces for the client and the resource applications and avoided any complexities. In this section, we demonstrate the execution of the proposed system through the help of several screen shots.

Once co-location server bootstrap the system through broadcasting its public key and connectivity information, the user can find it by clicking the 'Search for Coloc Server' menu item (figure 11(a)) from the client or resource application. In order to register several devices in the system, we have simulated different kinds of devices (i.e. printers, laptops, desktop computers, mobile phones) through creating their XML-based device profiles and stored them locally. Then each device is registered with the co-location server just by clicking the 'Device Registration' menu item (figure 11(b)) and then providing its XML-based device profile.

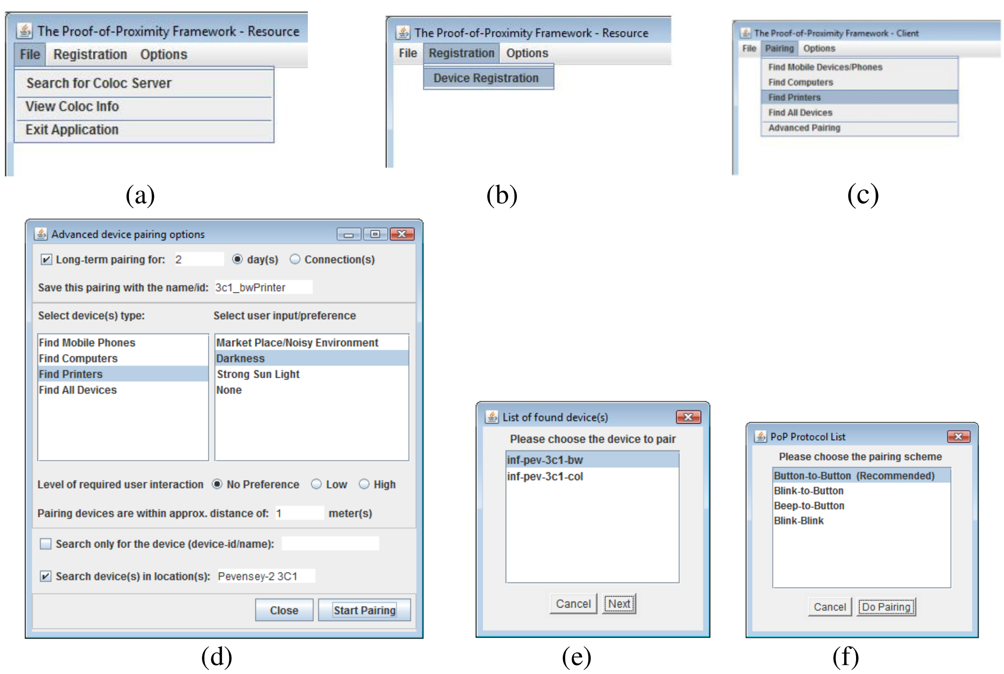

Figure 11: Screenshots demonstrating the prototype implementation

Once devices are registered with the co-location server, these can be discovered by the client application. User can use any one of the pre-set discovery options (figure 11(c)) or can use

'Advanced Pairing' menu item to perform an advanced device discovery and pairing (figure 11(d)). In advanced pairing user can establish a long-term pairing, optionally enter the user input/preference and location in which devices required to be searched. Based on the user's selection of the device-type (in this case printers), the client receives a list of the matching devices as illustrated in figure 11(e). User selects the intended device and clicks the 'Next' button to proceed. When user clicks the next button, another list box appears on the screen containing the list of PoP protocols in recommended order (figure 11(f)). Finally, the user selects the name of a PoP protocol and clicks the 'Do Pairing', which initiates the process of demonstrating the physical proximity of devices through the chosen PoP protocol.

5.3 Evaluation

As stated in order to evaluate the proposed system and to support our main argument that the integration of discovery mechanism and several proof-of-proximity protocols into a single device pairing system is an effective approach for ordinary users, we conducted a usability study. This is a study of the eight pairing schemes as well as the proposed system, which integrates them. The detail of the usability study is the scope of another paper; however the analysis of the results and the evaluation supports the assertion that the integration of the discovery mechanism and several proof-of-proximity protocols into a single system is a more effective approach to device pairing as compared to proposing and developing a plethora of pairing protocols that work in a totally independent fashion. In view of these facts, we believe that our work is an important and timely first step in academic research that highlights the need of a framework based approach to device pairing.

6. CONCLUSION

In this paper, we presented a framework based approach to device pairing. We showed the design and implementation of the proposed system and the CoLoc protocol, which is the core part of the proposed system. However, implementation results along with the details of the usability study are out of the scope of this paper due to the space limits. As future work, there are two extensions that can make it a more effective approach to device pairing. Firstly, the current prototype utilizes a co-location server in order to store and manage the devices profiles; however it is also possible that the system could be implemented without the co-location server, in which case the devices are responsible for maintaining the directory. Alternatively, a directory-less implementation of the proposed system is also possible in which case an out-of-band software component is required that should be responsible for secure exchange of the devices' capabilities. Secondly, the device registration and discovery process can be standardized through using existing standards, such as CC/PP [49] for describing device profiles and discovery queries.

ACKNOWLEDGEMENTS

This research was sponsored/funded by University of Sindh, Jamshoro, Pakistan under Mega Project Phase-I: No.SU/PLAN/F.SCH/650 and the work presented in this paper is the extended version of the conference paper published in FIT-2010 proceedings [1].

REFERENCES

[1] Malkani, Y.A., D. Chalmers, and I. Wakeman. A Framework for Secure Device Pairing by Demonstration of Physical Proximity. in ACM proceedings of Frontiers of Information Technology (FIT-2010).

[2] Stajano, F., The Resurrecting Duckling - What Next?, in Revised Papers from the 8th International Workshop on Security Protocols. 2001, Springer-Verlag.

[3] Stajano, F. and R. Anderson, The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks, in Security Protocols. 2000. p. 172-182.

[4] Stajano, F. and R. Anderson, The Resurrecting Duckling: security issues for ubiquitous computing. Computer, 2002. 35(4): p. 22-26.

[5] Naik, P., K. Ravichandran, and K.M. Sivalingam, Cryptographic key exchange based on locationing information. Pervasive and Mobile Computing, 2007. 3(1): p. 15-35.

[6] Kindberg, T., K. Zhang, and N. Shankar. Context authentication using constrained channels. in Proceedings of Fourth IEEE Workshop on Mobile Computing Systems and Applications. 2002.

[7] Saxena, N., et al., Secure Device Pairing based on a Visual Channel. in IEEE Symposium on Security and Privacy. June, 2006. Berkeley/Oakland, CA.

[8] Saxena, N., M.B. Uddin, and J. Voris. Universal Device Pairing using an Auxiliary Device. in Symposium On Usable Privacy and Security (SOUPS). 2008.

[9] Saxena, N. and J. Voris. Pairing Devices with Good Quality Output Interfaces. in International Workshop on Wireless Security and Privacy (WISP) (co-located with ICDCS). 2009.

[10] Prasad, R. and N. Saxena. Efficient Device Pairing using Synchronized "Human-Comparable" Audiovisual Patterns. in Applied Cryptography and Network Security (ACNS). 2008.

[11] Saxena, N. and M. Uddin, Automated Device Pairing for Asymmetric Pairing Scenarios, in Information and Communications Security. 2008. p. 311-327.

[12] Soriente, C., G. Tsudik, and E. Uzun. BEDA: Button-Enabled Device Association. in International Workshop on Security and Spontaneous Interaction (IWSSI 2007). 2007.

[13] Holmquist, L.E., et al., Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts, in Proceedings of the 3rd International Conference on Ubiquitous Computing. 2001, Springer-Verlag: Atlanta, Georgia, USA.

[14] Castelluccia, C. and P. Mutaf, Shake Them Up!: A Movement-based Pairing Protocol for CPU-constrained Devices, in Proceedings of the 3rd International Conference on Mobile systems, Applications, and Services. 2005, ACM: Seattle, Washington.

[15] Mayrhofer, R. and H. Gellersen, Shake Well Before Use: Authentication Based on Accelerometer Data, in 5th International Conference on Pervasive Computing (Pervasive 2007). 2007.

[16] Mayrhofer, R. and H. Gellersen. Shake well before use: two implementations for implicit context authentication. in Adjunct Proceedings of Ubicomp'07. 2007. Innsbruck, AT.

[17] Shaked, Y. and A. Wool. Cracking the Bluetooth PIN. in Proceedings of the 3rd International Conference on Mobile systems, Applications, and Services (MobiSys '05). 2005. Seattle, Washington: ACM.

[18] Kirovski, D., M. Sinclair, and D. Wilson, The Martini Synch: Using Accelerometers for Device Pairing. Technical Report MSR-TR-2007-123, Microsoft Research. September 2007.

[19] Soriente, C., G. Tsudik, and E. Uzun. HAPADEP: Human Asisted Pure Audio Device Pairing. Cryptology ePrint Archive, Report 2007/093.

[20] Buhan, I., et al. Secure Ad-hoc Pairing with Biometrics: SAfE. in Proceedings of First International Workshop on Security for Spontaneous Interaction (IWSSI ’07). 2007. Innsbruck, Austria.

[21] Balfanz, D., et al. Talking to strangers: Authentication in Adhoc Wireless Networks. in Symposium on Network and Distributed Systems Security (NDSS '02). 2002. San Diego, California.

[22] Nicholson, A., et al., LoKey: Leveraging the SMS Network in Decentralized, End-to-End Trust Establishment, in Pervasive Computing. 2006. p. 202-219.

[23] Buhan, I., et al., Feeling is Believing: A Location Limited Channel Based on Grip Pattern Biometrics and Cryptanalysis. Advances in Biometrics, 2007.

[24] Spahic, A., et al., Pre-Authentication using Infrared. Privacy, Security, and Trust Within the Context of Pervasive Computing, 2005. Vol. 780: p. 105-112.

[25] Mayrhofer, R., M. Hazas, and H. Gellersen, An Authentication Protocol using Ultrasonic Ranging: Technical Report. 2006, Lancaster University.

[26] Mayrhofer, R. and M. Welch. A Human-Verifiable Authentication Protocol Using Visible Laser Light. in the 2nd International Conference on Availability, Reliability and Security (ARES'07). 2007.

[27] Gehrmann, C. and C.J. Mitchell, Manual Authentication for Wireless Devices. RSA Cryptobytes, 2004. Vol. 7(1): p. 29–37.

[28] Varshavsky, A., et al., Amigo: Proximity-Based Authentication of Mobile Devices, in Ubiquitous Computing (UbiComp'07). 2007. p. 253-270.

[29] McCune, J.M., A. Perrig, and M.K. Reiter, Seeing-is-Believing: Using Camera Phones for Human-Verifiable Authentication. in IEEE Symposium on Security and Privacy (SP'05). 2005. p. 110 - 124.

[30] Ringwald, M. Spontaneous Interaction with Everyday Devices Using a PDA. in Proceedings of the Workshop on Supporting Spontaneous Interaction in Ubiquitous Computing Settings, (Co-located with Ubicomp02). 2002. Gothenburg, Sweden.

[31] Goodrich, M.T., et al. Loud and Clear: Human-Verifiable Authentication Based on Audio. in Proceedings of 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06). 2006.

[32] James Kempf and P.S. Pierre, Service Location Protocol for Enterprise Networks: Implementing and Deploying a Dynamic Service Finder, ed. C.A. Long. 1999, Canada: John Wiley & Sons, Inc.

[33] Miller, B.A. and C. Bisdikian, Bluetooth Revealed. 2nd edition. 2001: Prentice Hall PTR, Upper Saddle River, NJ, USA.

[34] Bluetooth SIG (Special Interest Group). 2010, URL:https://www.bluetooth.org/apps/content/.

[35] Universal Plug and Play (UPnP) Forum. 2010, URL: http://www.upnp.org/.

[36] The Community Resource for Jini Technology. 2010, URL:http://www.jini.org.

[37] Zhu, F., M. Mutka, and LionelNi, Classification of Service Discovery in Pervasive Computing Environments. Technical Report, Michigan State University, 2002.

[38] Bettstetter, C. and C. Renner. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol. in Proceedings of 6th EUNICE Open European Summer School (EUNICE-2000). 2000. Twente, Netherlands.

[39] Ververidis, C.N. and G.C. Polyzos, Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques. Communications Surveys & Tutorials, IEEE, 2008. 10(3): p. 30-45.

[40] E. Guttman, C.P., J. Veizades, M. Day, Service Location Protocol Version 2. June 1999, URL: http://tools.ietf.org/html/rfc2608, IETF RFC 2608.

[41] Droms, R., Automated Configuration of TCP/IP with DHCP. IEEE Internet Computing. 2(4): p. 45-53.

[42] Cai, T., et al. Simple Service Discovery Protocol. Internet Engineering Task Force (IETF), INTERNET DRAFT: draft-cai-ssdp-v1-01.txt, October 1999.

[43] Malkani, Y.A., D. Chalmers, and I. Wakeman, Secure Device Association: Trends and Issues, in Security of Self-Organizing Networks: MANET, WSN, WMN, VANET published in October 2010. ISBN: 978-1-4398-1919-7, A.-S.K. Pathan, Editor, Auerbach Publication.

[44] Malkani, Y.A., and L. Das Dhomeja. Secure device association for ad hoc and ubiquitous computing environments. in IEEE 5th International Conference on Emerging Technologies, ICET 2009. pg. 437-442.

[45] Kumaran, S.I., Jini Technology: An Overview. 2001: Prentice Hall PTR, Upper Saddle River, NJ, USA.

[46] Phidgets: Products for USB Sensing and Control. 2010.

[47] Oracle Berkeley DB XML, (2010). Available from: http://www.oracle.com/database/berkeley-db/xml/index.html.

[48] Mimaroglu, S. Java Programming with Berkeley DBXML, (2010). Available from: http://linux.sys-con.com/node/175405.

[49] Reynolds, F., et al., Composite Capability / Preference Profiles (CC/PP): A User Side Framework for Content Negotiation, W3C NOTE-CCPP-19990727, July 1999, url: http://www.w3.org/TR/NOTE-CCPP/

[50] Malkani, Y. A., A Proof-of-Proximity Framework for Device Pairing in Ubiquitous Computing Environments, PhD Thesis, School of Informatics, University of Sussex, UK, 2011.

Authors

Dr. Yasir Arfat Malkani is a Lecturer at the Institute of Mathematics and Computer Science (IMCS), University of Sindh, Jamshoro, Pakistan. He got his Master's degree in Computer Science from University of Sindh, Jamshoro (Pakistan) in 2003 and PhD from University of Sussex, Brighton, UK in 2011. His main area of research is Pervasive Computing. His research is focused on secure device/service discovery and access control mechanisms using policies and location/proximity data/information. He is also interested in sensor networks, wireless networks (including WiFi, Bluetooth, WiMAX, etc), and solutions to various issues in distributed and pervasive computing systems through the integration of tools and techniques from distinct disciplines/areas.

Dr. Dan Chalmers is a senior lecturer in the School of Informatics at the University of Sussex. Before working at the University of Sussex he worked for Imperial College London and Ericsson Ltd. He has a B.Eng (Hons) in Software Engineering from UMIST (now part of Manchester University), an MSc in Advanced Computing and a PhD both from the Dept. of Computing, Imperial College London. His research is focused on the way knowledge of context (including resource limits, location, and other physical and social aspects of context) can be used to modify behaviour, affect data display and configuration of systems.

Dr. Ian Wakeman is a senior lecturer in the School of Informatics at the University of Sussex. He has a BA in Electrical and Information Sciences from Cambridge University, a MS from Stanford University and a PhD from UCL. His research could be described as user-centred networking, investigating protocols and techniques to make computer networks work for people. This has spawned over 50 refereed papers in fields as diverse as congestion control for packetized video, programming languages for active networks and has more recently focused on trust based approaches for network and system configuration in pervasive computing.

Dr. Lachhman Das Dhomeja is an Assistant Professor at the Institute of Information & Communication Technology (IICT), University of Sindh, Jamshoro, Pakistan. He got his Master's degree in Computer Technology from University of Sindh, Jamshoro (Pakistan) in 1991 and PhD from University of Sussex, UK in 2011. His main research area is Pervasive Computing in general and policy-based context-awareness in particular. His other research interests include secure device pairing in ubiquitous environments, software architectures and Distributed Computing.