IntuiSec: A Framework for Intuitive User Interaction with Security in the Smart Home

by

Saad Zafer Shakhshir

B.S., Massachusetts Institute of Technology (2004)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2006
[February 2007]

Author . . . .
Department of Electrical Engineering and Computer Science
Sept 5, 2006

Certified by . . . .
Srini Devadas
Professor
pervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

IntuiSec: A Framework for Intuitive User Interaction with Security in the Smart Home

by

Saad Zafer Shakhshir

Submitted to the Department of Electrical Engineering and Computer Science
on Sept 5, 2006, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents IntuiSec, a framework for intuitive user interaction with Smart Home security. The design approach of IntuiSec is to introduce a layer of indirection between user-level intent and the system-level security infrastructure. This layer is implemented by a collection of distributed middleware and user-level tools. It encapsulates system-level security events and exposes only concepts and real-world metaphors that are intuitive to non-expert users. It also translates user intent to the appropriate system-level security actions. The IntuiSec framework presents the user with intuitive steps for setting up a secure home network, establishing trusted relationships between devices, and granting temporal, selective access for both home occupants and visitors to devices within the home. The middleware exposes APIs that allow other applications to present the user with meaningful visualizations of security-related parameters and concepts. I present the IntuiSec system design and an example proof-of-concept implementation, which demonstrates the user experience and provides more insight into the framework.

Thesis Supervisor: Srini Devadas
Title: Professor

Acknowledgments

First I would like to acknowledge my parents, Zafer and Hiba, for their unconditional support. And my sisters who are there for me always. Without them I would not have come this far.

My friends, who are fortunately too numerous to be named here, I thank you for being who you are - no more and no less.

I thank my supervisor Srini Devadas and my colleague at Nokia - Dimitris Kalofonos. I appreciate all the effort you put into this research and I hope this piece of work lives up to both of your expectations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Increasing numbers of network-capable devices and an unabated growth in global inter-connectivity are marking a clear path toward pervasive, networked computing. This, combined with the fact that the rate of malware infections has been not only rising, but constantly accelerating over the past two decades shows that there is a clear need to critically readdress the design of secure systems.

Traditionally, computer security has been a high priority only for entities dealing with extremely sensitive data, such as the military, the government, and banks. Information Technology (IT) experts in each individual branch or institution were able to configure and manage their own internal security mechanisms. However as network-capable consumer devices become more powerful and prevalent, average consumers are growing more reliant on these devices for transactions involving increasing amounts of personal information - such as bank account and credit card details.

Yet despite this, most users are still perplexed by the cumbersome experiences of configuring and managing their application, device, and network security settings. One particular domain where this issue is being brought to the forefront is that of wireless security. Estimates vary however most analysts seem to agree that 60 to 70 percent of home wireless networks have no security at all [1]. This situation will only exacerbate as the number of home networks grows rapidly from 37 million worldwide in 2003 to a projected 111 million in 2008, according to market advisory firm IDC [2].

Concomitant to this increase in wireless proliferation is a tangible movement towards ubiquitous computing whereby users are surrounded with a multitude of network-capable devices that gather information, provide services, and communicate with each other. This will take place even within individual homes transforming what is currently a simple home network consisting of a handful of devices into a substantially more complex *Smart Home.* If the average user finds the security of today's relatively simple home networks arduous to configure and maintain, then surely the arrival of the Smart Home will substantially add to the burden.

1.1 A Different Approach to User Interaction with Security

This significant change in status quo requires a paradigm shift in the design of secure systems for user-oriented environments. The average consumer cannot be expected to interact with security in the ways that IT experts traditionally have and most current systems are not designed with non-expert users in mind. Recently, some attempts have been made to improve this situation. However many such attempts involve improving the graphical user interface (GUI) while still forcing the user to interact directly with low-level security parameters, such as Wired Equivalent Privacy (WEP) keys and Medium Access Control (MAC) address filters. These attempts are inadequate as they still directly expose average users to complicated underlying security mechanisms.

Rather than redesigning the lower level cryptographic protocols, which are necessarily complex and accomplish their goals of (among other things) securing and verifying data, I argue that there is a need for a level of abstraction between system-domain security functionality and user-level actions and intent. This abstraction must be bi-directional, as shown in Figure 1-1. It must encapsulate system-level security events in such a way that comprehensible notifications can be presented to the user and it must also do the reverse by translating user-level intent into the appropriate

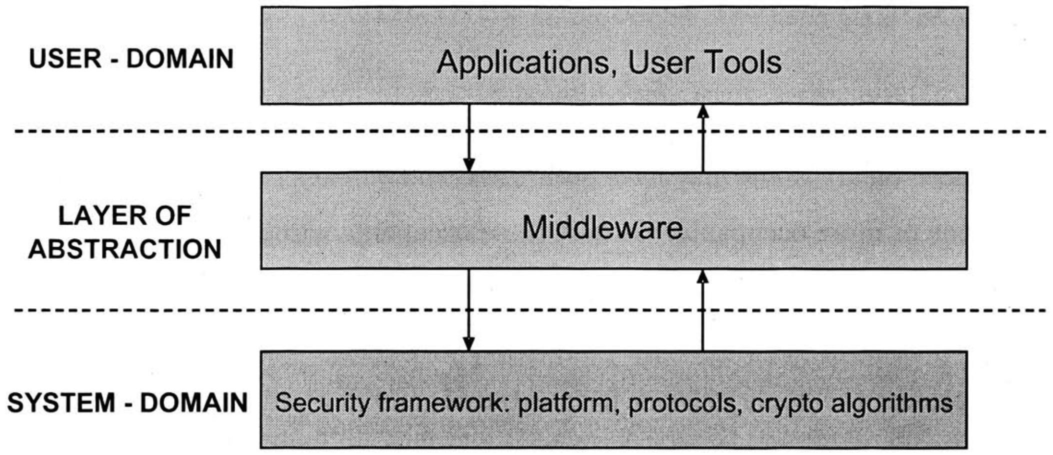


Figure 1-1: Layer of abstraction between system-domain security and user-domain applications

system-level actions. Furthermore, the abstraction must be applicable across a variety of possible underlying security mechanisms so as to provide a consistent user experience when interacting with security.

To demonstrate adoption of this strategy, I present a framework that accomplishes the aforementioned goals for a specific application domain. The framework is named *IntuiSec* (INTUItive SECurity) and is based on research done at both MIT and the Nokia Research Center (NRC) [3] [4] [5]. It consists of a middleware, user-level tools, and lower-level security mechanisms. The user tools use Application Programming Interfaces (APIs) exposed by the middleware to present the user with a consistent and comprehensible interaction with the underlying security mechanisms. These APIs are meant to help applications provide *implicit security* whereby low-level security functionality is hidden from the user and the applications can focus on the user's goals rather than forcing the user to deal with complicated security functionality. Furthermore, the framework creates a separation of policy from mechanism since the underlying security mechanisms may change whilst the higher level policies may remain the same. Note that no changes are made to the underlying cryptographic algorithms. IntuiSec uses widely adopted security mechanisms, such as public key cryptography, symmetric encryption, and hashing algorithms.

1.2 Application Domain and Motivation

IntuiSec is designed with a specific application domain in mind – the Smart Home. The IntuiSec Smart Home may have both wired and wireless connectivity and it may contain one or more occupants. Each of these occupants wishes to protect his or her own devices from unauthorized use and wants to be able to grant selective, temporal use of their devices to both home residents and visitors. Collectively all occupants want to protect the home network from external unauthorized connectivity. The IntuiSec framework presents the user with intuitive steps for bootstrapping new devices onto the home network, establishing trusted relationships between devices, and granting temporal, selective access for both home occupants and visitors to devices within the home. The middleware exposes APIs that allow other applications to present the user with meaningful visualizations of security-related parameters and concepts. The framework does not impose restrictions on exactly how such concepts are presented, although the implementation described in this thesis demonstrates how the APIs may be leveraged to display meaningful representations.

A motivating example of this application domain can be seen in the initiatives being proposed by the Digital Living Network Alliance (DLNA). This large consortium of companies over a variety of technology industries is standardizing the interoperability of wired and wireless devices in the home. They have already released several specifications that are now being incorporated into consumer devices [6].

Furthermore, the use of mobile devices plays a central role in the user interaction processes described in this thesis. Specifically, cell phones have become the most widespread consumer device with over 2 billion subscribers in 2005 and a projected 3.2 billion in 2010 [7]. This means that not only are they pervasive, but they are also personal, meaning an individual does not usually share regular use of his mobile phone with another person. Mobile phones also have significant storage and computational capabilities, with support for short range wireless and near-field communication options. This combination of features makes them prime candidates for use as control devices within Smart Homes. Several key aspects of the IntuiSec framework are designed with this in mind.

1.3 Contributions

This thesis introduces several novel approaches in addressing the issue of user interaction with security frameworks in Smart Home environments:

- Creation of a bi-directional layer of indirection to translate between user-level intent and system-level security actions in the Smart Home. This layer of indirection is not restricted to a specific set of underlying security mechanisms - it exposes consistent API's despite varying lower level security protocols.

- Introduction of abstracted security tickets known as *Passlets* which are user-perceived entities that encapsulate access control parameters to be distributed from one user to another. Passlets incorporate both connectivity-level security information and device-level authorization parameters thereby allowing configuration on both levels to take place in one easy step.

- Creation of a key-rotation mechanism for connectivity-level wireless security whereby access points rotate shared cryptographic secrets regularly and devices are given an ordered sequence of valid secrets. The length of the distributed sequence is determined by the duration of authorized connectivity allowing for automatic termination of connectivity access without the need for manual revocation.

- Implementation of a working system that demonstrates the framework's functionality on the Series 60 mobile phone platform and the Linux platform within a simulated Smart Home environment.

1.4 Organization

This thesis begins with a summary of related work in Chapter 2. Chapter 3 gives introductory information needed in order to appreciate the remainder of the thesis.

Following this in Chapter 4 I give a detailed description of the design of the framework. In that section I also explain the security model that applies to IntuiSec. Chapter 5 describes my implementation of the IntuiSec framework on Nokia Series 60 cellular phones and Linux laptops in a simulated smart home environment. Finally, in Chapter 6 I conclude and offer suggestions for further work.

Chapter 2

Related Work

IntuiSec is the result of research in three main disciplines: Security, Human Computer Interaction (HCI), and Smart Spaces. While there are many publications in each field individually, it is more meaningful and relevant for the purposes of this thesis to look at work that fits into some combination of the three. Thus I will begin by describing publications on user interaction with security, followed by publications on user interaction with smart spaces, followed by publications on security in smart spaces, and finally I will present publications covering all three disciplines.

2.1 User Interaction with Security

One approach in this domain has been to critically examine the usability of existing applications and technologies with respect to security. In [8], Whitten and Tygar perform a rigorous analysis on Pretty Good Privacy (PGP) and conclude that while it was intended to provide a global framework for securing and authenticating emails, it did not meet their usability requirements. A study was conducted on users in which they found that only three out of twelve test subjects successfully managed to use PGP to encrypt and decrypt email. Similar conclusions are reached in [9] where Zurko et. al. examined user behavior when presented with security question pop-ups during workflow while using the Lotus Notes application. They conclude that almost half of the users make the 'wrong choice' and let the insecure content run. The Kazaa peer-to-peer (P2P) file sharing application is looked at in [10] and found to fail in its handling of privacy related functionality. Users inadvertently share information and files that they would like to keep private.

Other researchers have focused on more general aspects of security usability rather than examining specific applications. In [11], Dourish et. al. conducted a series of qualitative interviews at both an academic institution and an industrial research lab. They found that "neutral to negative attitudes dominated the end-user experience of security technologies" and whilst not focusing on ubiquitous computing as such, they recognize the increased complexity and urgency of creating usable security frameworks for pervasive computing. Yet more papers examine the relationship between usability and security in an attempt to create general design guidelines [12][13].

A separate series of studies from University College in London explores the interaction between users and passwords highlighting the deficiencies of password-based systems and making recommendations for improvement [14][15]. As a result, several other designs using alternative forms of authentication have been suggested [16][17].

Researchers tend to agree that usability needs to be factored into the design of a system from the start. One interesting question raised in the literature is whether security can and should be retro-fitted onto existing systems that have proven to be usable, or whether a system should be designed with security in mind from the start. In [18], Gutmann et. al. argue the case for post hoc security noting that most successful secure applications were previously insecure, such as Skype, Gnutella, Bit Torrent, and even Secure Shell (SSH) which added encrypted connections onto the existing user experience of the telnet application. Others also highlight the success of SSH in providing implicit security whereby a user takes application-level actions that are automatically coupled with security actions [19]. In a rare positive comment on existing solutions, they mention technologies that they believe are "getting it right", such as the identity-based encryption scheme by Boneh and Franklin whereby one can encrypt messages using public strings like email addresses [20].

Another point that researchers make is the importance of presenting the user with appropriate visualizations in order to make informed decisions related to security [21].

DiGioia and Dourish take a novel approach in this regard leveraging social navigation to present more meaningful visualizations to the user [22].

Finally there is some work that presents more concrete attempts at creating applications with a more usable interaction with security, but these are more recent. One such example is SPARCLE, a "set of privacy utilities that are intended to assist organizations with the creation, implementation, and internal auditing of privacy policies" [23].

Besides the above academic approaches, the issue of security usability has attracted a lot of interest in the industry, especially in the domain of wireless networking. There are two main initiatives worth noting: Buffalo Technology's AirStation OneTouch Secure System (AOSS) [24] and Linksys' SecureEasySetup (SES), which was adopted from Broadcom [25]. Both technologies promise quick and easy setup of secured wireless networks and have similar user interfaces despite using different underlying security mechanisms. Currently there is an effort by the Wifi Alliance to standardize this process using non-proprietary means.

2.2 User Interaction with Smart Spaces

There are several papers that give an overview of current research relating to smart spaces and present the various challenges that designers face when creating products for these environments [26][27][28]. The 'Intelligent Room' is one such environment which uses artificial intelligence (AI) techniques with video cameras and microphones to perform facial recognition, speech recognition, and gesture tracking [29]. On the other hand in [30], Connelly and Khalil present a framework whereby spaces and devices communicate with each other to perform policy resolution. Devices are then automatically configured to adhere to the resolved policy. Katsuno and Aihara take a different approach by separating a device into a mobile core and a fixed cradle. They propose an autonomic network configuration technology that enables these mobile cores to discover information about the network and configure the settings accordingly [31].

2.3 Security in Smart Spaces

Again in this field there are several papers that given an overview of current research [32][33]. Campbell et al. take this one step further in [34] by also presenting their own prototype implementation to address some of these issues. One approach to security in smart spaces is to use a set of environment variables to create an environment role which then dictates the access control to various resources [35]. Another framework has been proposed to manage key distribution at the link level [36]. Both these approaches use a centralized server.

2.4 User Interaction with Security in Smart Spaces

The overall scope of this thesis involves work in all three fields - usability, security, and smart spaces. Hence research that combines these three areas is the most relevant. However it is also the most rare. In [37], the authors present a framework that uses trust levels in order to establish access control. Users are assigned levels of trust by the appliances. The description of how this is done is quite high level and it does not address security on the lower connectivity levels.

Others leverage location limited channels (LLC's) as part of their frameworks. In [38], Balfanz et al. describe their "Network in a Box" which uses the infrared port of a laptop to auto-configure the laptop's wireless network security settings. Another illustrative example is given in [39] where LLC's are used to exchange public key information between a laptop and a printer in an airport lounge. This then allows the laptop to securely print to that specific printer over SSL or TLS. They follow up on this research in [40] where they describe five usability lessons learned from conducting an experiment with two different methods of setting up a public key infrastructure (PKI) based wireless network. The first method involved users performing manual configuration of their devices and the second involved using the infrared method described earlier. Their results showed that it took users an average of 140 minutes to configure their devices manually and only 1 minute and 39 seconds using the LLC-based auto-configuration method. All the participants were computer literate, typically with a PhD in computer science.

Holmström takes a different approach in [41] using the metaphor of a business card to delegate permissions between individuals. The example given there is granting your colleague permission to read your email while you are away. The framework presented uses Simple PKI certificates as these do not rely on a hierarchy of certification authorities thereby addressing the scalability problems of traditional X.509 identity certificates.

Lastly, it is important to describe work done by the Universal Plug and Play (UPnP) Forum on home network security. The forum consists of a large number of companies across a wide range of technology industries and is strongly supported by DLNA which has adopted their standards for interoperability within the Smart Home. The security aspect is one component of the larger UPnP Device Architecture that was standardised and adopted by the Forum in 2003. The security framework proposed by the Forum is specific to the Smart Home environment. It introduces a *Security Console* which is a software component that handles the 'client-side' of the security protocols. These Security Consoles are the identity-bearers in the home and not users. The *Device Security* component handles the 'server-side' of the security interaction. The framework uses existing notions of certificates to grant access to devices. It however does not account for granting of connectivity access and the different connectivity mechanisms that a device may use to join a home network. Their proposals also do not focus on the framework's usability, which is clearly a key element of any Smart Home network infrastructure. Their work is best outlined in [42]. For a more detailed technical description, see [43], [44], and [45].

Chapter 3

Design Rationale

3.1 Security Model and Threats

IntuiSec is intended to operate in a Smart Home environment. An example of a Smart Home is a house with both wired and wireless connectivity that contains numerous devices providing various services to users in the home. Such devices could be media servers, audio/visual (A/V) renderers, refrigerators – or any household appliance or consumer electronics device, as depicted in Figure 3-1. Currently many of these devices are not yet network-capable, but the technology to make them so is available and there are various initiatives strongly advocating for such adoption in the near future. For an example of such an initiative, see the Digital Living Network Alliance (DLNA) at [6].

Smart Homes are likely to have several occupants. IntuiSec assumes that each occupant would like to protect the privacy of her own devices whilst being able to grant selective access to other individuals as she sees fit. Collectively as a home, IntuiSec assumes that the occupants would like to prevent unauthorized non-home occupants from gaining any access to the home network; this includes even connectivity access, such as the ability to acquire an IP address and passively read network traffic. IntuiSec also works under the assumption that home occupants would like to be able to grant visitors access to devices within their homes for temporary periods of time.

Thus the threats that IntuiSec aims to protect against are:

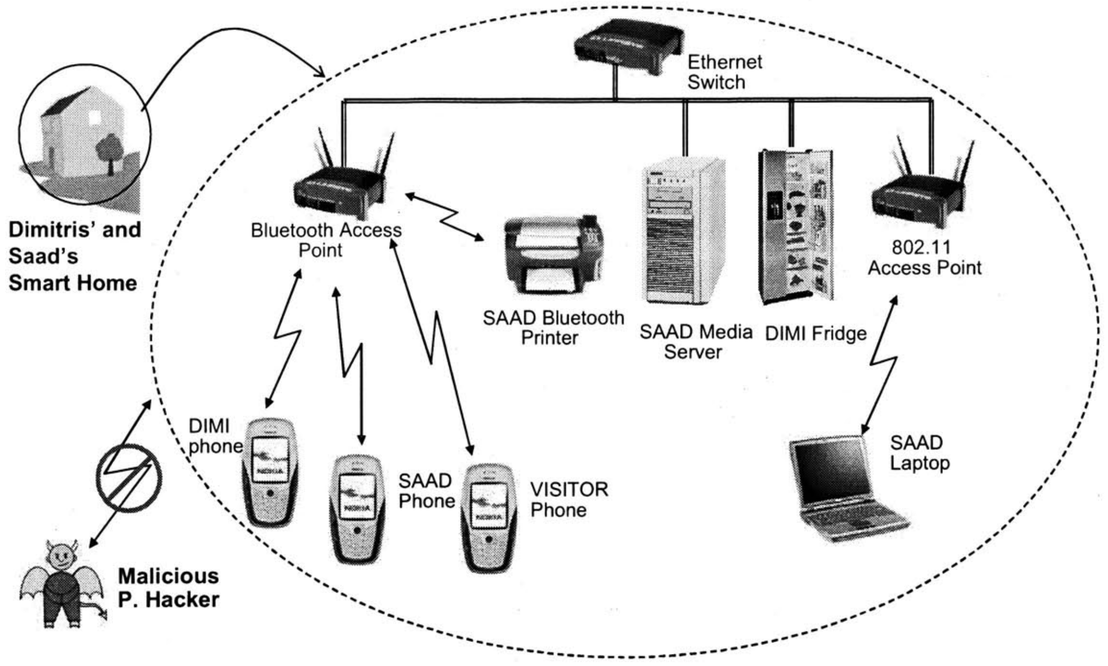


Figure 3-1: An Example of the IntuiSec Smart Home

1. Unauthorized connectivity access to the home network through any means – Bluetooth, 802.11, etc.

2. Unauthorized access to services on a device in the home by anyone that has not been granted permission to access it by the owner of that device

To address these threats, the access control provided by IntuiSec takes place on two levels:

1. **Connectivity Level** – prevents unauthorized devices from connecting to the home network

2. **Service / Device Level** – prevents already connected users from accessing device functionality for which they are not authorized and allows users to reliably authenticate the devices they are interacting with.

In terms of protecting against unauthorized connectivity access, IntuiSec focuses on wireless connectivity. Gaining illicit wired access is seen as a much less significant threat to home network security than gaining access over a wireless connection since the intruder must have physical access to the home in order to connect over a wire – i.e. they must break into a person's home before they can attach a malicious device with a wire. A device that is attached by wire is also more noticeable than a person snooping over a wireless connection and so could more easily be detected and removed. Furthermore, if an intruder does happen to acquire unauthorized wired connectivity to the home, he or she will still not be able to access the services offered by any of the home devices due to the second layer of protection at the service level.

IntuiSec is not intended to protect a household against the most advanced adversary. Its aim is to provide a strong level of protection for average users in a smart home environment and to do so in an intuitive manner. The framework presents a shift in the design of secure systems and with this it hopes to both decrease the number of unprotected wireless home networks and to provide a foundation for smart home security frameworks. It does not aim to redesign existing cryptographic algorithms; it leverages them as part of the framework. IntuiSec assumes platform security when storing sensitive data on devices and it trusts manufacturers to ensure that their devices are not pre-shipped with malicious software.

3.2 Use Cases

This section presents several basic use cases for user interaction with security in the Smart Home. The scenarios are presented in the context of a household with two occupants - Saad and Dimitris.

The first use case involves bootstrapping new devices into the Smart Home. When Saad or Dimitris purchases a new device, they would like it to obtain permanent connectivity to the Smart Home network over a secure channel. At the same time they would like to protect their home from being accessed by non-authorized users. Figure 3-2 illustrates this. The process for achieving this would ideally be straightforward, consistent, and intuitive, which is clearly not the case with today's methods of dealing WEP keys, MAC address filters, etc.

Once the devices are connected and can all communicate with each other securely,

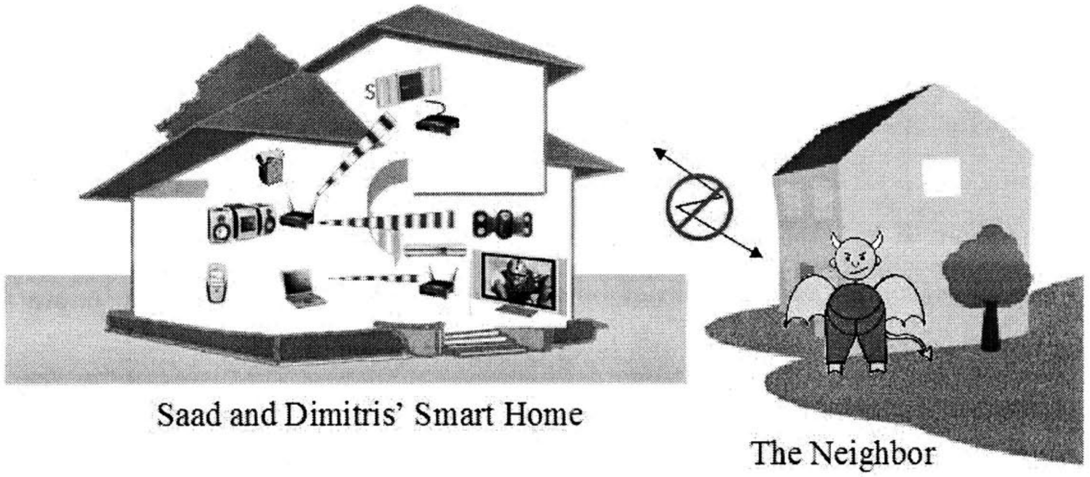
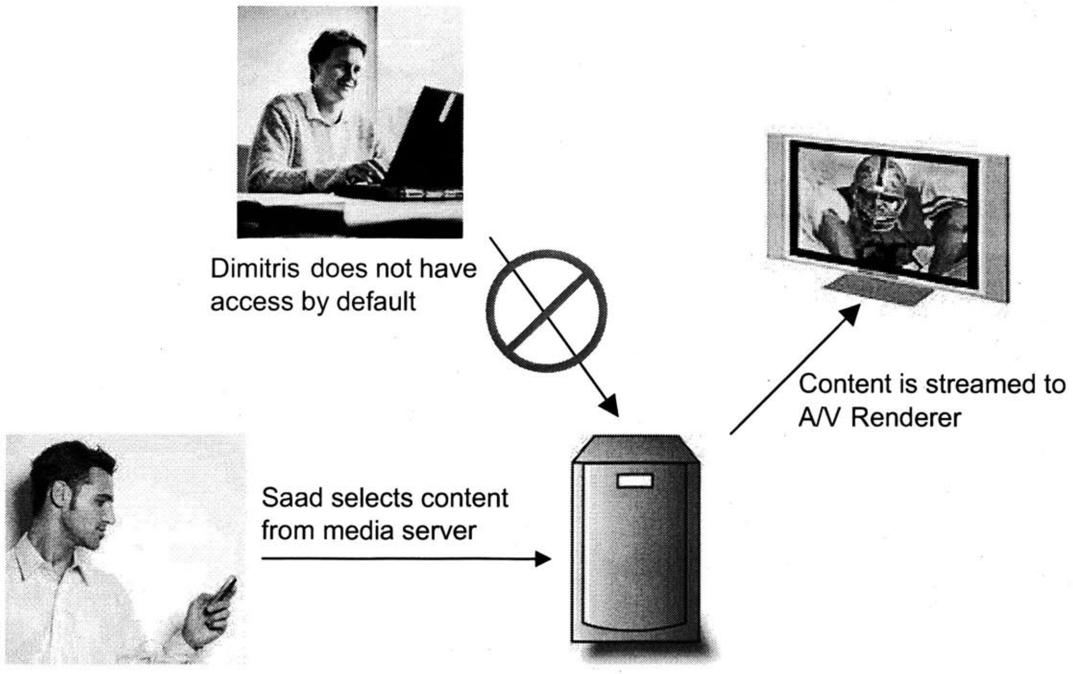


Figure 3-2: The Secure Smart Home

Saad would like to prevent Dimitris from accessing his devices until he explicitly grants him access. He could also opt to have his devices have some default level of access to everybody. By way of example, Saad purchases a new media server whose content he wishes to keep private. Both Saad and Dimitris jointly purchased the A/V renderer in the living room so they both have access to it by default. Thus Saad can now stream content from his media server and display it on the A/V renderer, however Dimitris cannot. Figure 3-3 illustrates this.

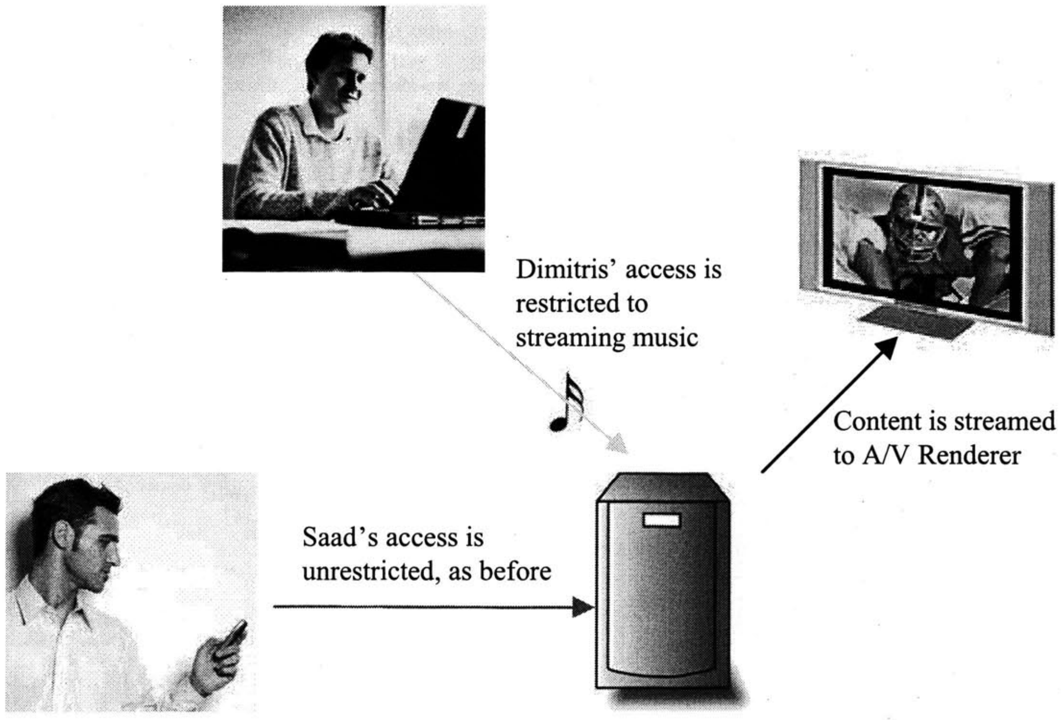
Dimitris later decides that he wants to play some music from the server and asks Saad to give him access. Saad agrees, but he only wants to give Dimitris permission to stream music and not movies from the server. He does not want him to access any other file types or to upload or delete anything. Figure 3-4 illustrates the situation after Saad grants Dimitris limited access to his media server.

A further use case involves granting temporary access to visitors for use of specific services in the Smart Home. Again, by way of example, the fridge in Saad and Dimitris' home breaks down. The repairman comes over but by default is not even able to connect to and browse their home network. However, Saad grants the repairman access for the day to selected functionality provided by the fridge so that he can perform the necessary repairs. At the end of the day, once the work is complete, the repairman automatically loses all access to Saad and Dimitris' Smart Home network.



Dimitris does not have access by default
Content is streamed to A/V Renderer
Saad selects content from media server

Figure 3-3: Before Saad grants Dimitris access to his media server



Dimitris' access is restricted to streaming music
Content is streamed to A/V Renderer
Saad's access is unrestricted, as before

Figure 3-4: After Saad grants Dimitris access to music on his media server

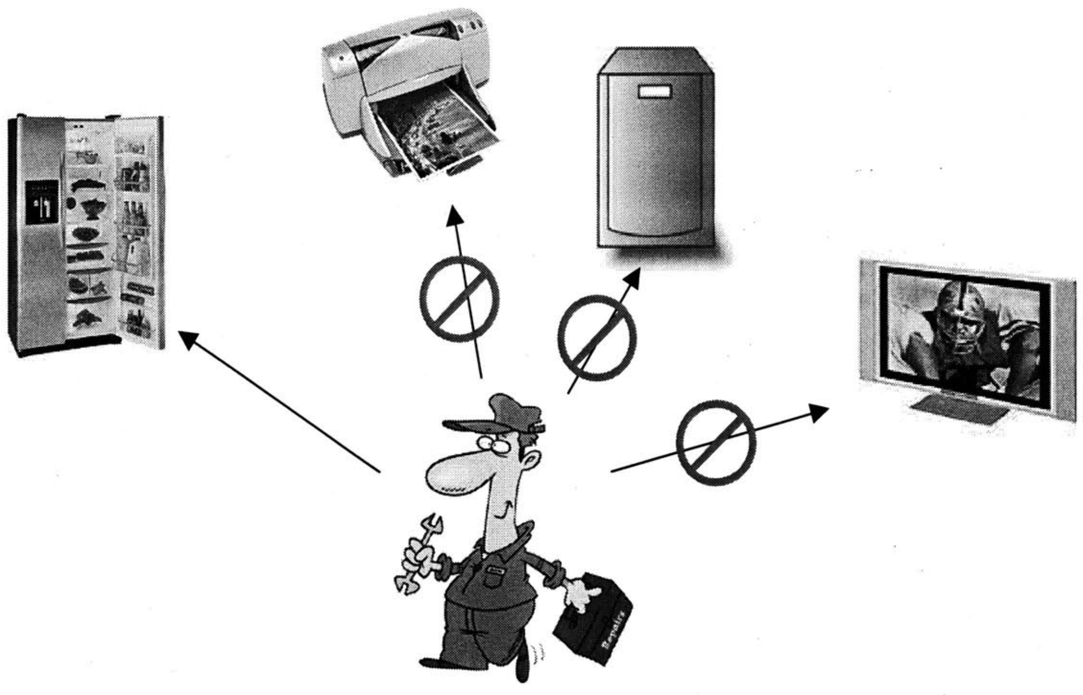


Figure 3-5: The repairman's access while fixing the fridge

Figure 3-5 shows the repairman's access during the day.

3.3 Design Goals

The overall design goal of IntuiSec is *to provide a usable framework through which non-expert users can securely create and manage complex Smart Home environments regardless of the underlying security mechanisms.* In this light, the framework should have the following functionality:

Bootstrapping – the framework should enable users to easily add new devices into the Smart Home without the need to be directly exposed to a variety of tedious and counter-intuitive security mechanisms.

Authenticating – the framework should enable users to easily establish trusted relationships between devices. This is similar to the concept of trusted websites on the Internet.

Controlling Access – the framework should enable users to easily provide other users with various levels of access control to devices that they own. This access may be temporary and the user should not be forced to manually revoke the permissions when the desired access time has expired. From the user's perspective, the process of granting these permissions should be identical regardless of whether the recipient is a home occupant or a visitor. However visitors should also be automatically given the necessary connectivity parameters to connect securely over wireless connections to the Smart Home. Devices that belong to home occupants do not require the connectivity security parameters since they will have already obtained them through the *Bootstrapping* process.

Visualizing Security – the framework should enable users to visualize the various security relationships and parameters by abstracting them to fit into concepts that they are already familiar with. For example, users are already familiar with the concept of keys to unlock the front door to their home. It should be feasible to incorporate such concepts into the framework.

3.4 Assumptions

IntuiSec makes certain assumptions. These are listed and discussed here:

- *There exists some form of near field communications (NFC) channel on every device.* This is a strong assumption, however with the decreasing costs and increased proliferation of RFID technology, it is fair to assume that a large number of devices will incorporate some type of NFC capability in the near future.

- *Communications over NFC's are very difficult (though not impossible) to eavesdrop on.* It is very difficult for an adversary to capture information transferred over NFC's due to the short range of communication (e.g. a few centimeters for passive RFID tags). In addition, given the home environment in which IntuiSec is designed to function, this scenario is even more unlikely.

- *Communications over NFC's are authentic.* Any information obtained over an NFC channel is authentic information coming from the device in question. In the unlikely case that an adversary attempts to manipulate this connection, it will be readily detectable.

- *Platform Security.* This assumption states that each device's on-board software is responsible for maintaining the integrity and privacy of data stored on the device. This means that information used by IntuiSec is stored in restricted areas of the filesystem that only authorized users may access.

- *Out-of-the-box hardware and software are trustworthy.* IntuiSec does not aim to prevent against malware that comes pre-shipped with either device hardware or software.

- *Some distributed middleware with a request/response framework exists.* IntuiSec does not provide functionality for service discovery or device manipulation over the network. However it assumes such a framework exists in the context of the Smart Home and that such a framework can leverage IntuiSec for its security requirements.

Chapter 4

Design

4.1 Design Concepts and Nomenclature

Before progressing further, it is important that the reader understands certain concepts and terminology that are used in the remainder of this thesis. I define them in this section for easy reference.

4.1.1 Device Types

There are several different types of devices in the IntuiSec Smart Home. **Home Devices** are devices which have common knowledge of the **Home Secret**. This secret is given to a Home Device when the user 'unlocks' the home network for it. The procedure for unlocking the home network is part of the Easy Setup process described in Section 4.2 and may be performed using a physical object reminiscent of a key. The key that may be used is named the **PhyKey** (a PHYsical KEY as opposed to the virtual keys used in cryptography).

There are several properties that all Home Devices share:

- Home Devices have permanent connectivity access to the home network.
- Home Devices are the only devices that can be used to allow **Visitor Devices** to access the home network and to selectively use some of the home services

Home Devices are further divided into three different types:

1. **Infrastructure Devices**, such as Access Points (APs) and gateways. These devices provide network connectivity to other devices within the home.

2. **Mobile Devices**, such as cellular phones and Personal Digital Assistants (PDAs). These devices can be easily moved around and can therefore be used to touch all three types of Home Devices.

3. **Fixed Devices**, such as fridges and microwaves. These devices cannot be easily moved around and so cannot be used to touch other Fixed Devices or Infrastructure Devices. A mobile intermediary can be used to touch one Fixed Device and then another in order to securely transfer information between them, as is done during the Easy Setup process described in Section 4.2.

Another important distinction to make between the various Home Devices is whether or not a device is used to *control* and access specific services on the network, or whether it is *providing* such services. Although not part of the framework, IntuiSec assumes the existence of some underlying service discovery and control protocols. An example of such a suite of protocols is given by the UPnP architecture, which has been chosen by DLNA as the standard for interoperable device communication within the Smart Home. I leverage this particular framework as part of the implementation of IntuiSec described in Chapter 5. For more information on UPnP, see [46].

A device used to control other devices is named a **Control Point** and a device that provides services to the network is named a **Provider**. The same physical device may be both a Control Point and a Provider. An example of this relationship is a user using his cell phone as a Control Point to download content from a media server, which is a Provider. Note that UPnP uses the term 'Device' to describe a Provider, however I use the latter in this thesis as it reduces ambiguity.

4.1.2 User Roles

The first user of a brand new device becomes its first **Owner**. This happens through the Easy Setup process described in Section 4.2. Owners, by default, have full access to any of the services on a device. Only an Owner can do the following:

- Grant permission to other users to access services offered by the device.
- Add other users to the device's Owner list

Users interact with Providers in the home through Control Points. When a user logs onto a Control Point, any subsequent network requests are issued on behalf of the currently logged on user. Therefore, it is users and not devices that are the initiators of actions in the IntuiSec Smart Home. This enables a user's network privileges to persist regardless of which device or Control Point she is using.

Users can be either **Home Occupants** or **Visitors**. Home Occupants are Owners of at least one Home Device. They can selectively grant access to other Home Occupants or Visitors for services on the devices that they own. Visitors are Owners of Visitor Devices. These devices do not have knowledge of the Home Secret and so do not have connectivity access to the home network by default. Visitors can only be granted access to the home network and its services by a Home Occupant.

4.1.3 Secrets, Identities, and Cryptographic Keys

Each Home Device stores the **Home Secret**, which has the following properties:

- the **HomeID** is defined as the hash of the Home Secret and it is used to uniquely identify the Smart Home ($HomeID = hash(Home\ Secret)$).
- The Home Secret may be distributed using the PhyKey or by other means such as a manual input method. For a discussion of the PhyKey see section 4.2.2.

Each user is assigned a system-wide private/public keypair $\{UK_{PRI},\ UK_{PUB}\}$ with the following properties:

- $\{UK_{PRI}, UK_{PUB}\}$ may be generated using different mechanisms, such as a username/password pair or biometric information. The most important property of the keypair is that it is consistent throughout the system so that a user can be properly identified regardless of the device she is using. A form of identity based encryption (IBE), as described in [22] may also be used here to generate a keypair.
- the **UserID** is defined as the hash of UK_{PUB} ($UserID = hash(UK_{PUB})$) and it is used to uniquely identify each user in the framework.
- the UserID of the Owner of a device denotes that device's **OwnerID**

Each Provider has a unique private/public keypair $\{PK_{PRI}, PK_{PUB}\}$ that is pre-shipped with the device. This keypair has the following property:

- the **ProviderID** is defined as the hash of PK_{PUB} ($ProviderID = hash(PK_{PUB})$) and it is used to uniquely identify Providers in the framework.

4.1.4 Touch Authentication Process (TAP)

Touching is a concept used extensively by the IntuiSec framework. The term **TAP** is used to describe the user action of using a Mobile Device to touch another device. The Mobile Device being used is known as the **TAPing Device** and the device that is subsequently touched is known as the **TAPed Device**. During a TAP, the system leverages some form of NFC to perform any of several security-related transactions, such as the exchange of public keys, mutual verification of the Home Secret, transferal of cryptographic secrets, etc. A Mobile Device may be used as an intermediary to perform a TAP between two non-mobile devices, such as during the Easy Setup process described in Section 4.2.

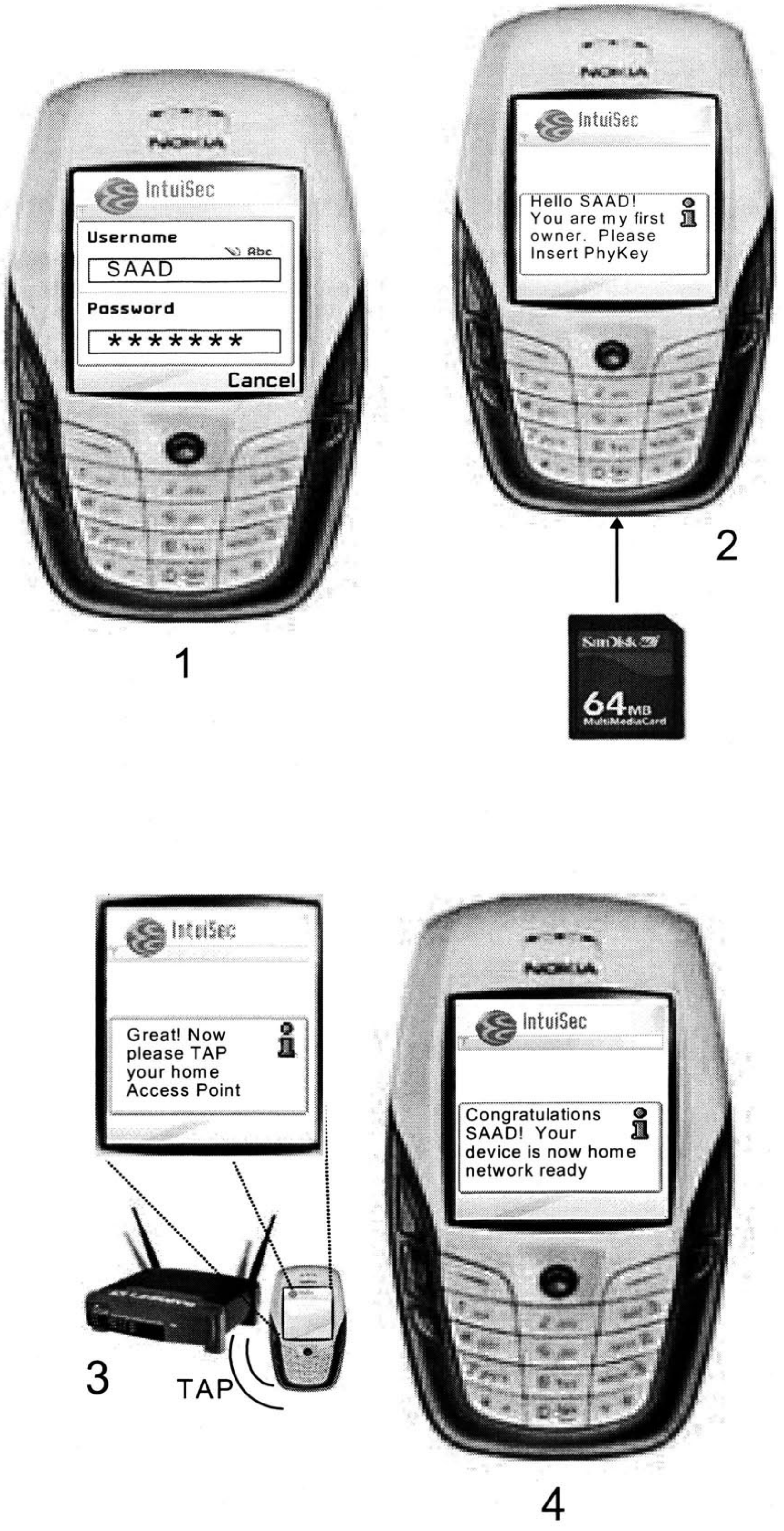
4.2 Easy Setup

IntuiSec aims to provide a "buy-plug-and-play" user experience as much as possible in order to reduce the burden of security on users. When a user purchases a new device and wishes to bootstrap it for secure home networking, he performs three simple steps. Figures 4-1 and 4-2 depict this process pictorially. Below is a more thorough description of the process and a discussion highlighting several significant concepts.

Step 1: The user performs the imprinting process, by which the new device comes to learn about its first Owner [47]. Figures 4-1 and 4-2 show a username/password entry system, however this is not a requirement and the Owner may be imprinted through other means such as a fingerprint reader, retina scanner, or any other method that can uniquely identify a person. It is essential that this information be consistent throughout the user's interaction with the system so that their identity remains constant. The framework uses this user-specific information to independently generate the user's key pair $\{UK_{PRI}, UK_{PUB}\}$, where $Hash(UK_{PUB}) = OwnerID$ of the device.

Step 2: The user 'unlocks' the home network by transferring the Home Secret to the device. This can be done with the insertion of the PhyKey, as shown in Figures 4-1 and 4-2, however the use of a PhyKey is not a requirement. It could be that the user is prompted to input the Home Secret manually. See Section 4.2.2 for the rationale behind using the PhyKey

Step 3: The final step of the Easy Setup process requires the user to TAP an Infrastructure Device in order to acquire the necessary link level security parameters to connect to the Home Network. If the TAPing device is a Home Device, transfer of the connectivity security parameters is then done over a location limited channel using some form of NFC, such as infrared or RFID. If the device being set up is a Fixed Device, then an already bootstrapped mobile intermediary may be used to perform this third step by TAPing the Fixed Device as in Figure 4-2. If the device being set up is an Infrastructure Device then this step is not required and the device may simply be plugged in to the home network. As part of the TAP in this step, the device that is transmitting the connectivity security information verifies that the device being set up is a Home Device by prompting it to prove that it knows the Home Secret. There are several ways



Hello SAAD! You are my first owner. Please Insert PhyKey

Great! Now please TAP your home Access Point

Congratulations SAAD! Your device is now home network ready

Figure 4-1: Easy Setup of a Mobile Device (e.g cellular phone)

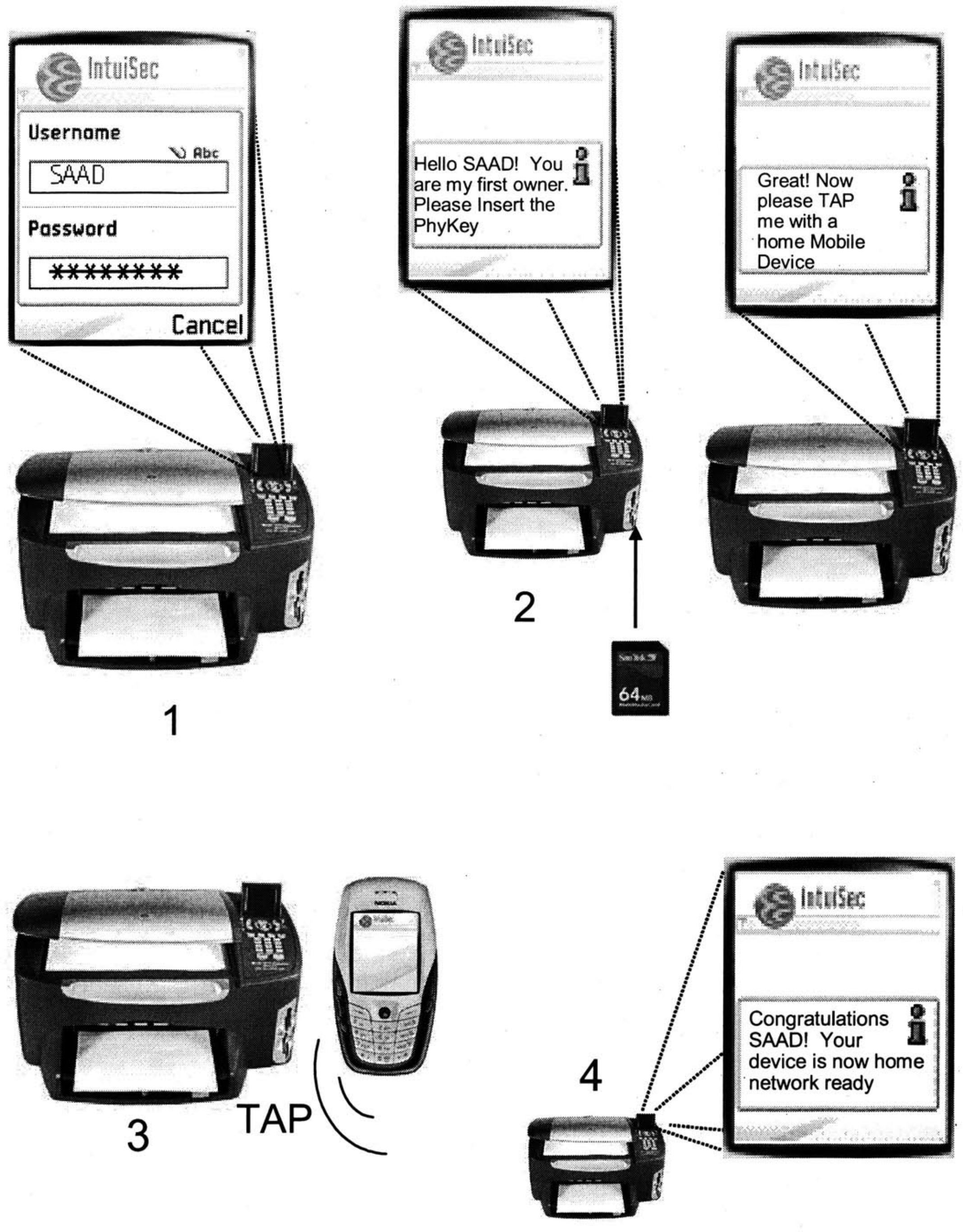

IntuiSec
Username
Abc
SAAD
Password
****
Cancel
1
IntuiSec
Hello SAAD! You are my first owner. Please Insert the PhyKey
2
64MB
IntuiSec
Great! Now please TAP me with a home Mobile Device
3
TAP
4
IntuiSec
Congratulations SAAD! Your device is now home network ready

Figure 4-2: Easy Setup of a Fixed Device (e.g Bluetooth printer)

to do this, however a straightforward challenge-response protocol such as the following is sufficient: the transmitting device can use the Home Secret to encrypt a random sequence of bytes known as a *nonce* and then send that nonce to the device that is being set up. It is then up to the device being set up to decrypt the nonce using the same Home Secret as a shared key and then send back its plain-text form to the other device for verification.

After completion of the Easy Setup process, the newly set up device can be used to securely connect to the home network. All bootstrapped Home Devices share the following properties:

- All connections over wireless links will have connectivity-level security. Devices that have not performed the Easy Setup process will not be able to connect by default.
- Home Devices can disconnect and reconnect with no extra configuration requirements
- Service and device discovery are now possible over the network
- All devices have at least one owner

4.2.1 Ordered List of Secrets and Key Rotation Mechanism

IntuiSec introduces a key rotation mechanism that improves both the usability and the security of existing wireless solutions. The idea here is to introduce a software layer that replaces the user interaction needed for creating and changing common secrets or encryption keys in Smart Home environments. By automatically setting the user-level secrets, this method removes the burden from non-expert users and makes the configuration easy for them. At the same time it also becomes applicable to a wide variety of underlying security systems, all of which support the concept of a user-level common secret. This is illustrated in Figure 4-3. It also allows for time delimited access control privileges that expire without the need for manual revocation.

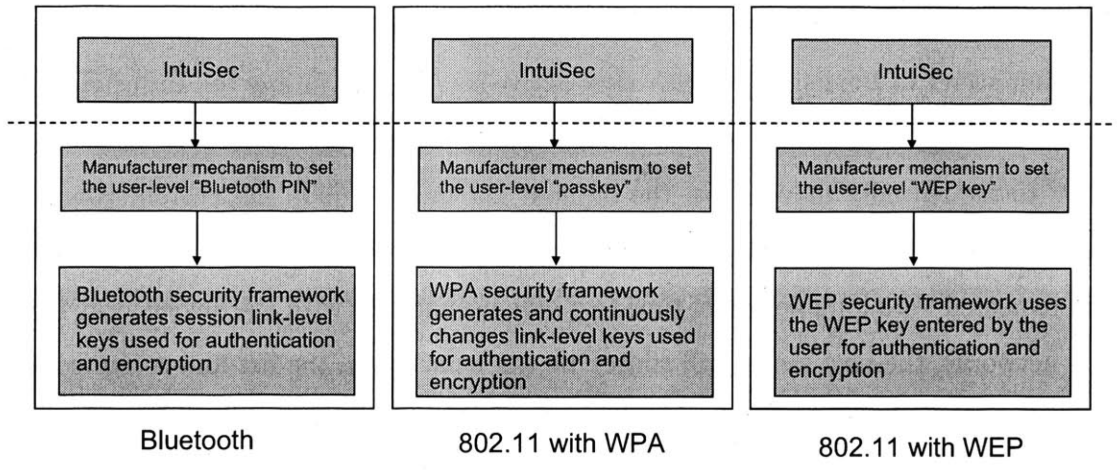

IntuiSec
IntuiSec
IntuiSec
Manufacturer mechanism to set the user-level "Bluetooth PIN"
Manufacturer mechanism to set the user-level "passkey"
Manufacturer mechanism to set the user-level "WEP key"
Bluetooth security framework generates session link-level keys used for authentication and encryption
WPA security framework generates and continuously changes link-level keys used for authentication and encryption
WEP security framework uses the WEP key entered by the user for authentication and encryption
Bluetooth
802.11 with WPA
802.11 with WEP

Figure 4-3: An illustration of how the ordered secrets mechanism interacts with different wireless link-level security implementations

To accomplish this, IntuiSec introduces an algorithm to set and update user-level common secrets based on an ordered list mechanism. According to this mechanism, an ordered list of user-level common secrets is generated by the IntuiSec software running on an Infrastructure Device. This list can be generated randomly using a seed selected by the manufacturer of the device or entered by the user in the form of a password or by using an RFID tag, etc. The device then selects the first secret in the list and uses the manufacturer's selected mechanism to set the user-level common secret with it. The Infrastructure Device periodically selects the next secret in the ordered list and again sets the user-level common secret with it. When the Infrastructure Device reaches the end of the ordered list, it generates a new list either automatically based on a seed selected by the manufacturer or by prompting the user for a new password, a new RFID tag etc. This is expected to happen infrequently because it is very feasible to have lists containing, for example, 1000 secrets or more (e.g. with secrets of 16 bytes, the list would require 16 KB memory, which is readily available today even in very inexpensive hardware). This means that even changing a secret every day, the ordered list will be exhausted once approximately every 3 years in this example. The process of renewing the entire ordered list could take place at any time if, for example, the members of the Smart Home feel that the list has been compromised.

During Step 3 of the Easy Setup process, this list of ordered secrets is transferred to the device that is being set up. The newly setup device then stores this list and sets the first secret in the list as the currently active secret. Every time the Infrastructure Device changes the common secret, the other device momentarily loses access and will have to sequentially move down the ordered list until it finds the current common secret used by the AP in order to successfully reconnect. Unless the client device stays disconnected for a long period of time (e.g. the user travels with the device away from the network), the new secret will simply be the next one in the list and reconnection will take place almost instantly. If it has exhausted the list of secrets without finding the correct one, then the device can correctly assume that the Infrastructure Device has progressed beyond the list of secrets that it obtained when it performed the Easy Setup. It then notifies the user that a TAP is necessary to obtain a new list of secrets.

This scheme also allows users to easily grant temporary access to the Smart Home network to visitor client devices. For example, if a user wants to give a visitor access for three days and the Infrastructure Device is configured to change the common secret every day, then IntuiSec transfers to the Visitor Device the current and the two following secrets in the ordered list. This process is automatically incorporated into the Passlet mechanism described in Section 4.4

4.2.2 Use of the PhyKey

Using a PhyKey in Step 2 of the Easy Setup process has several advantages over manual password input methods. The first advantage is its ease of use: the user does not need to know anything about the contents of the PhyKey. The user only needs to know that inserting the PhyKey is part of what is required to make a device a Home Device and that other devices which have not received the contents of the PhyKey will not have permanent access to the home network.

The second advantage is that it is more flexible and secure. The contents of the Home Secret can be anything since IntuiSec has total control over what these contents are. One could argue that by having the user manually enter a password instead of using the PhyKey, IntuiSec could then use that password as some type of seed to generate the Home Secret. However this would restrict the number of possible

Home Secrets to a relatively small and finite amount since passwords that are easy to remember come from a subset of characters, numbers, and perhaps punctuation. The more complex the password, the harder it is for the user to remember.

In addition, the PhyKey also provides an analogy to existing keys that people use for their doors in the sense that the PhyKey 'unlocks' the home network, just like the house key unlocks the front door. Since people are already accustomed to actions such as unlocking their doors, they should find it easier to adopt these actions as common practice when it comes to securing their home networks.

However, the issues that apply to someone losing the key to their home also analogously apply to someone losing the PhyKey. If the PhyKey is lost and the home occupants are worried that it fell into the hands of someone who knows where their home is, a new one would have to be created with a different Home Secret. This will prevent future intrusion by whoever is in possession of the lost PhyKey. Nevertheless IntuiSec envisions that the PhyKey is placed in a drawer somewhere and is only seldom taken out in order to bootstrap a new device into the home. Also, backup copies of the PhyKey may be made just like copies of regular keys may be made. However, to further protect against threats caused by losing the PhyKey, an additional step is needed to finalize the Easy Setup process for both Mobile and Fixed Devices and this step requires the user to be in physical proximity of a Home Infrastructure Device in order for them to TAP it (step 3).

4.3 Trust Builder

Once the Home Devices are connected to the home network, they can proceed to search for and discover each other. But how can the user trust devices that she has discovered over the network using her Control Point? A device that claims to be Saad's Printer may not in fact be the device it is claiming to be.

The concept of building trust between devices is used to allow the user to credibly verify the identity of a device that she is communicating with over the network. When browsing the web, certificates are used to enable a client to verify the identity of a remote server. When a client connects to a server, it requests a certificate from the server in order to perform authentication. This certificate should be signed by one of the trusted root Certificate Authorities (CA's) that come pre-installed with the client web browser. The browser can then check the validity of this certificate by verifying its signature against that of one of the root CA's that it knows about. For a more detailed discussion on CA's, see [48].

In order to eliminate the need for CA's in the Smart Home, IntuiSec leverages the inherent high level of security that NFC's provide compared with long distance network connections. When communicating with a remote server, the user may be easily tricked into communicating with a malicious middleman acting as the remote server. This is known as a "Man-in-the-Middle Attack". However when communicating over a location-limited channel, this threat is virtually non-existent, as described earlier in Section 3.4.

NFC's also facilitate the use of touch as an intuitive method of establishing trusted relationships with devices. The premise is that it is easier for a user to trust what she has seen and touched, not something that she has never seen.

As also discussed in Section 3.4, IntuiSec assumes manufacturers do not ship compromised hardware to the consumer. IntuiSec also trusts that the hardware will maintain platform security. Given these assumptions and the elimination of the possibility of a Man-in-the-Middle Attack, IntuiSec considers information received over NFC channels to be trusted information. This means a Control Point can obtain trustworthy initial rendezvous information from a Provider through a TAP. The rendezvous information may consist of several pieces of data, such as the Provider's ProviderID, its OwnerID, and its HomeID. The network browsing application on the Control Point that is used to discover and control devices on the network can then use this information to display meaningful visualizations of these security parameters. These visualizations may incorporate whether or not a device is a trusted device by displaying a green background for trusted devices and a red background for devices that are not trusted. Again, IntuiSec does not dictate how such visualizations are presented, however it provides the infrastructure to facilitate such a display. An example of how this information may be presented is given in Figure 4-10.

4.4 Passlets

Most current schemes for granting access to a resource over a network require the user to perform some direct manipulation of access control configuration. The configuration data could be in the form of an Access Control List (ACL), a list of MAC addresses, or a number of other implementations. The mechanism could reside either on the device itself or on some centralized third party entity, such as an authentication or authorization server. Furthermore, the quasi-static state of the settings is often impractical for most common use cases, which involve granting of privileges for a temporary period of time.

The process of granting access is in general cumbersome for non-expert users. When user interaction is required, direct contact with security is not intuitive and a better GUI design alone is usually not the solution [8] [10]. For example, assume Alice comes over to Dimitris' and Saad's house and she wants to browse Saad's shared pictures on his media server using her laptop over 802.11 wireless. Dimitris and Saad, being security-conscious individuals, have protected their home network by employing a MAC filter and 128-bit WEP encryption on their wireless AP. To grant Alice connectivity access, Saad must first figure out what the MAC address of her laptop is. He then has to connect to the AP's configuration page and manually enter her laptop's MAC address into the "Allow" list. After that he must manually enter the 128-bit WEP key or passphrase onto Alice's laptop. Now at this point, Saad has only provided Alice with connectivity access to the wireless network and not access to his shared pictures. To grant her access to his pictures under any modern operating system such as Windows, OS X, or Linux, he has to log onto the device where the folder is located and manually edit the shared settings of that folder. This may involve even further complications such as the creation of an additional user account specifically for Alice on the device. In addition to all this, if Alice came back a few days later and secretly snooped around near Saad's house, she would most likely still be able to connect to his home network and to access his pictures unless he has manually revoked her access privileges. Clearly this is far from optimal.

As a major step in improving this scenario, IntuiSec introduces Passlets, which are *user-perceived* entities that carry *user-level* intent. They behave like 'tickets' or 'passes' that give permission to the bearer to connect to the home network and use a subset of services provided by a networked device, without requiring that device to be configured in advance to do so. The user needs to know nothing of the underlying security infrastructure, which in general can come in many different flavors. The emphasis is to enforce implicit security allowing the user to focus on their goals and to separate the mechanism of the underlying security infrastructure from the actions or the policy of the user. Saad's goal in the above example is to allow Alice to access his shared pictures for the duration of her visit. His goal is not to discover what the MAC address of her laptop is, nor is it to manually configure ACL's that have to be later on edited again in order to terminate Alice's access privileges.

Passlets are inspired by the concept of 'capabilities'. They allow users to intuitively grant access to other users to selectively use devices they own. The IntuiSec Passlet mechanism includes:

- A middleware that provides a level of indirection between the user domain and the security system domain. The middleware translates user-intent to system-specific actions.
- User level tools to capture the user intent. They allow users to generate, transfer, view and revoke Passlets for devices they own

4.4.1 User Interaction

To demonstrate the usability of Passlets, contrast the following steps which Saad performs using IntuiSec with the aforementioned prevailing method of granting Alice access to his shared pictures:

1. Using his cell phone or any other device that he owns, Saad browses through his list of owned devices, selects the media server, and specifies the appropriate

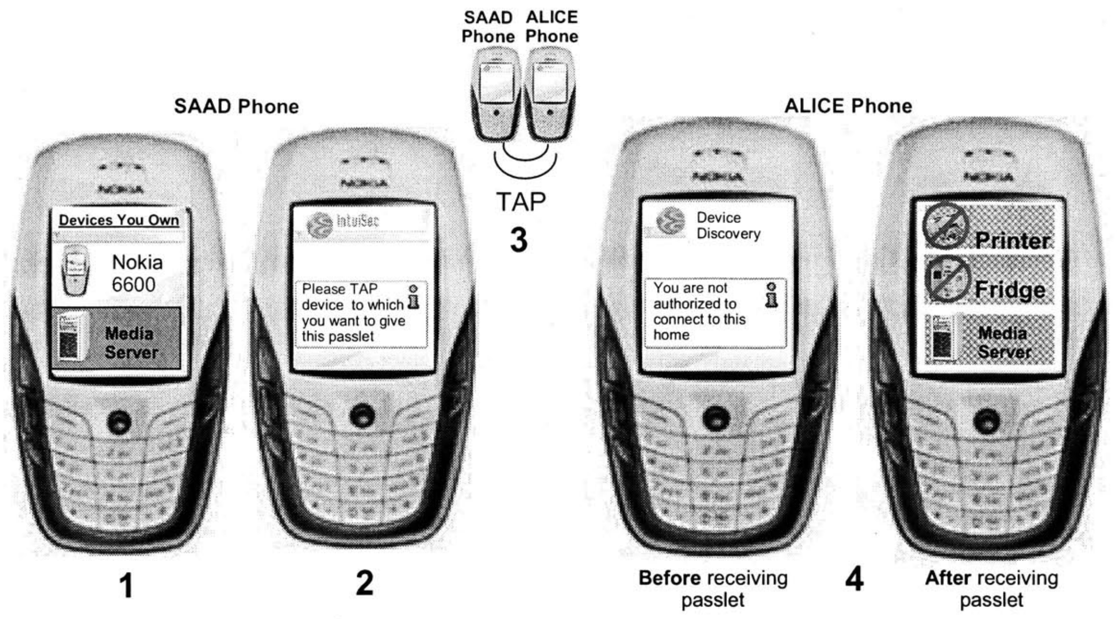


Figure 4-4: Saad using a Passlet to grant Alice temporary access to music on his media server

parameters for the Passlet. Note that these parameters are meaningful on a user-level, such as which folder he wishes to share, how long he wishes to share it for, what level of access he wishes to give (read or write), etc.

2. Saad TAPs Alice's laptop.

See Figure 4-4 for a depiction of these steps pictorially. Alice is shown using a mobile phone and not a laptop however the procedure is identical in either case highlighting the consistency of IntuiSec's user interaction.

4.4.2 Example Passlet Designs

The internals of Passlets will vary depending on the underlying security mechanisms that are being used and therefore IntuiSec does not dictate a single specific implementation of Passlets. However there are some general technical principles which are common to all forms. Passlets are first digitally signed using the private key of the owner of a device and then encrypted using the public key of the Provider (PK_{PUB}) for which the Passlet is being generated. This digital signature provides three levels of protection for the inner contents of a Passlet:

1. *Authenticity* – The recipient can verify that the sender is in fact who s/he claims to be.

2. *Integrity* – The recipient can verify that the message has not been altered during transmission.

3. *Non-repudiation* – The recipient can ensure that an identical message is not sent again by a third party.

With only digital signing however, the contents may still be viewed by a third party. To avoid this, a Passlet is encrypted with PK_{PUB} of the Provider that the Passlet is being generated for. This ensures additional privacy since only that Provider may view the contents of the Passlet.

I now present several different examples of how a Passlet may in fact be implemented. The first example illustrates how a Passlet destined for a home occupant may be implemented. The subsequent three examples describe Passlets that are granted to visitors. Passlets for visitors add an additional layer of complexity since connectivity-level security information must also be incorporated into the Passlet.

Example 1

This example describes a Passlet that is sent from one home occupant to another. This case is the simplest as both users already have the necessary security parameters to connect to the home network. All that needs to be transmitted here is information for access control to the Provider. A Passlet of this type is shown in Figure 4-5 followed by a description of its various parameters. As motivation for this type of Passlet, recall the use case where Saad wants to grant Dimitris access to stream music from Saad's media server.

UserID: The UserID of the user for whom the Passlet is being generated. This would be Dimitris' UserID in the example use case.

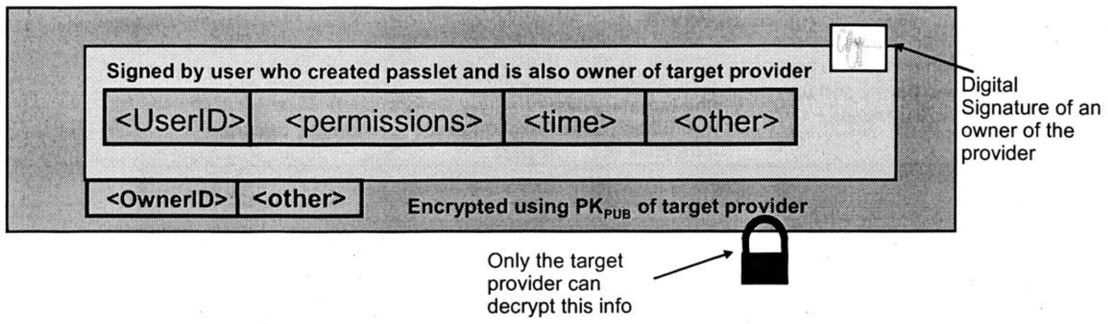

Signed by user who created passlet and is also owner of target provider
<UserID>
<permissions>
<time>
<other>
<OwnerID>
<other>
Encrypted using PK_PUB of target provider
Digital Signature of an owner of the provider
Only the target provider can decrypt this info

Figure 4-5: Passlet to be given to a home occupant

Permissions: The permissions that the recipient is being granted. This would be 'stream music files'.

Time: An expiry time or number of uses for this Passlet.

OwnerID: The OwnerID of the person generating the Passlet. In the example, this would be Saad's UserID. It is included so that if a Provider has multiple owners, then when it receives a Passlet it will be able determine which owner has generated the Passlet in order to verify the digital signature. To eliminate the need for this, a Provider could simply use trial and error with the public keys of all the owners to see which one verifies, however this is computationally more expensive and the OwnerID is not sensitive information as it is the hash of the public key of the owner.

Other: Possible extra information

Example 2

This first example of a visitor Passlet works without any modifications to existing out-of-the-box Bluetooth and 802.11 access points. The information that is sent in the clear consists of some public and some sensitive information, which is why it is preferred that this type of Passlet be transferred only through a secure mechanism such as TAPing. Recall that TAPing involves the simple touching of one device and another. A Passlet of this type is shown in Figure 4-6. Note that the inner portion of this Passlet is identical to Example Passlet 1, however it contains additional

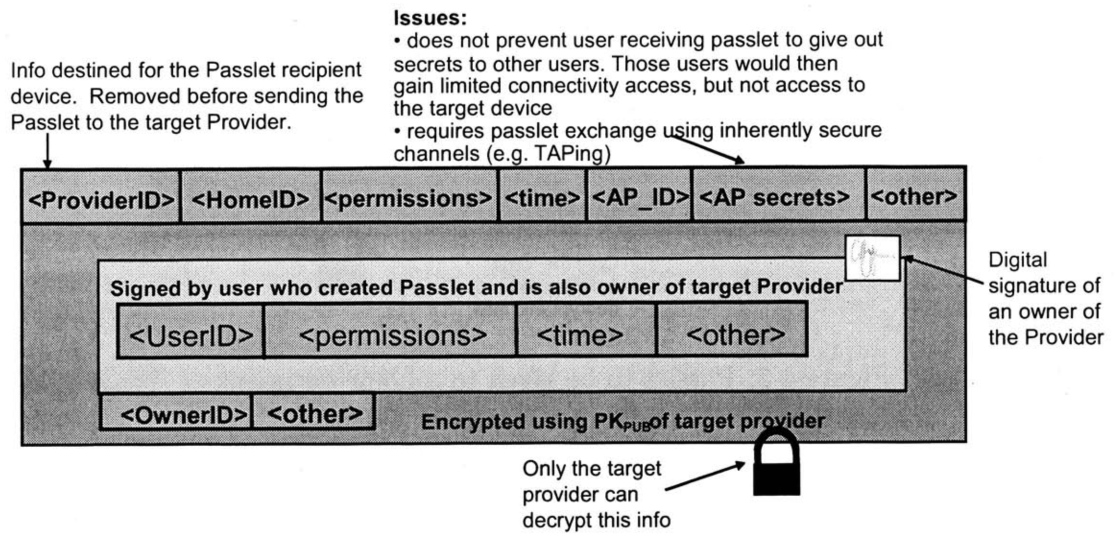

Issues:
• does not prevent user receiving passlet to give out secrets to other users. Those users would then gain limited connectivity access, but not access to the target device
• requires passlet exchange using inherently secure channels (e.g. TAPing)
Info destined for the Passlet recipient device. Removed before sending the Passlet to the target Provider.
<ProviderID>
<HomeID>
<permissions>
<time>
<AP_ID>
<AP secrets>
<other>
Signed by user who created Passlet and is also owner of target Provider
<UserID>
<permissions>
<time>
<other>
Digital signature of an owner of the Provider
<OwnerID>
<other>
Encrypted using PKPUB of target provider
Only the target provider can decrypt this info

Figure 4-6: Passlet for a visitor without requiring extra computation by the Access Point

information that is sent in the clear and these are described below. As motivation for this type of Passlet, recall the use case where Saad wants to grant Alice permissions to view his shared pictures on his media server for the day.

The information that is sent in the clear is:

ProviderID: This is the ProviderID of the Provider that the Passlet is providing access to. This would be the ProviderID of Saad's Media Server in the example use case.

HomeID: The HomeID of the home which the Provider is a part of.

Permissions: The permissions granted to the user. These are purely for the purposes of the user's device knowing what privileges it is being granted in advance so that any GUI's can be updated. These are identical to the signed permissions, which are the ones that are actually used by the Provider to enforce access control.

Time: The expiry time of the Passlet. Analogous to the situation with permissions, the time here is identical to the signed time and is only used by the recipient's device to inform the user when the Passlet is meant to expire.

AP_ID: The unique identifier of the connectivity access point. For Bluetooth and 802.11 access points this would be the hardware or MAC address.

AP secrets: These are the user-level secrets required to connect to the access point with the specified AP_ID. For Bluetooth this could be a sequence of one or more PINs that are used to establish a pairing between the device and the access point. For 802.11 this could be a sequence of one or more WEP passphrases.

If intercepted, the AP_ID combined with the AP secrets would allow an adversary to establish connectivity with the home network. However, due to the additional layer of protection, he would not be able to access any functionality at the Provider level. He would still be able to monitor packets and perhaps flood the network or cause a Denial of Service (DoS) attack. As such it is recommended that this type of Passlet be sent through a TAP rather than over a long-distance channel. This would nearly eliminate the chances of an adversary picking up the sensitive information.

Example 3

This type of Passlet, shown in Figure 4-7, adds an additional layer of protection on top of the previous Passlet by encrypting the entire Passlet with the public key of the user for whom the Passlet is destined. This would eliminate the threat of a passive adversary learning the AP secrets. However, if the recipient user is not trustworthy then she may decide to share these secrets with a potentially malicious third party after decrypting the outer portion of the Passlet. Since this Passlet is encrypted with UK_{PUB} of the user that the Passlet is for, it does not have to be sent through a TAP.

Example 4

This type of Passlet, shown in Figure 4-8, is more secure than the previous example since it encrypts its entire contents using the Home Secret. This means that the recipient cannot decrypt the Passlet and so does not have access to any AP secrets. However, this scheme requires a modified AP that supports Passlet decryption. When a visitor attempts to access a device within the home, her device sends the Passlet to

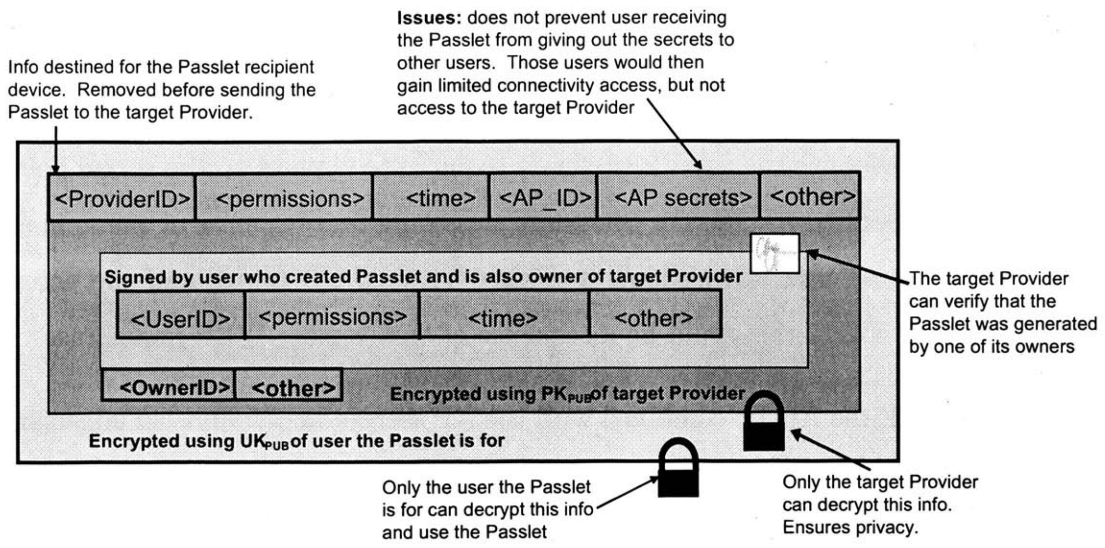

Info destined for the Passlet recipient device. Removed before sending the Passlet to the target Provider.
Issues: does not prevent user receiving the Passlet from giving out the secrets to other users. Those users would then gain limited connectivity access, but not access to the target Provider
<ProviderID>
<permissions>
<time>
<AP_ID>
<AP secrets>
<other>
Signed by user who created Passlet and is also owner of target Provider
<UserID>
<permissions>
<time>
<other>
The target Provider can verify that the Passlet was generated by one of its owners
<OwnerID>
<other>
Encrypted using PK$_{PUB}$ of target Provider
Encrypted using UK$_{PUB}$ of user the Passlet is for
Only the user the Passlet is for can decrypt this info and use the Passlet
Only the target Provider can decrypt this info. Ensures privacy.

Figure 4-7: Passlet for a visitor – encrypted so as to prevent eavesdropping

the AP which decrypts it and ensures that the Passlet is valid. This verification can be done by sending a Message Authentication Code (MAC) along with the Passlet in order to establish authenticity and integrity. If the Passlet is valid then the AP sends the user's device the current secret in order to connect to the home network over an encrypted channel. After connecting to the network, the device then proceeds to connect to the Provider in the same manner as in Example 3. A more intelligent AP may even decouple the user authentication mechanism from the encryption key. This way upon each subsequent attempt to connect, the AP authenticates the Visitor using his UserID and checks that against a list of valid Passlets. If the Visitor has a valid Passlet, then his device is sent the current secret and he is allowed to connect securely to the home network.

4.4.3 Passlet Sessions

Passlet sessions are initiated when the bearer of the Passlet does not have an already established Passlet session with a Provider and attempts to access its functionality. The process through which this occurs is shown in Figure 4-9. Note that after the initial unpacking and verification of the Passlet, shared session keys are created and used for encrypting future communications as this is a less computationally intensive

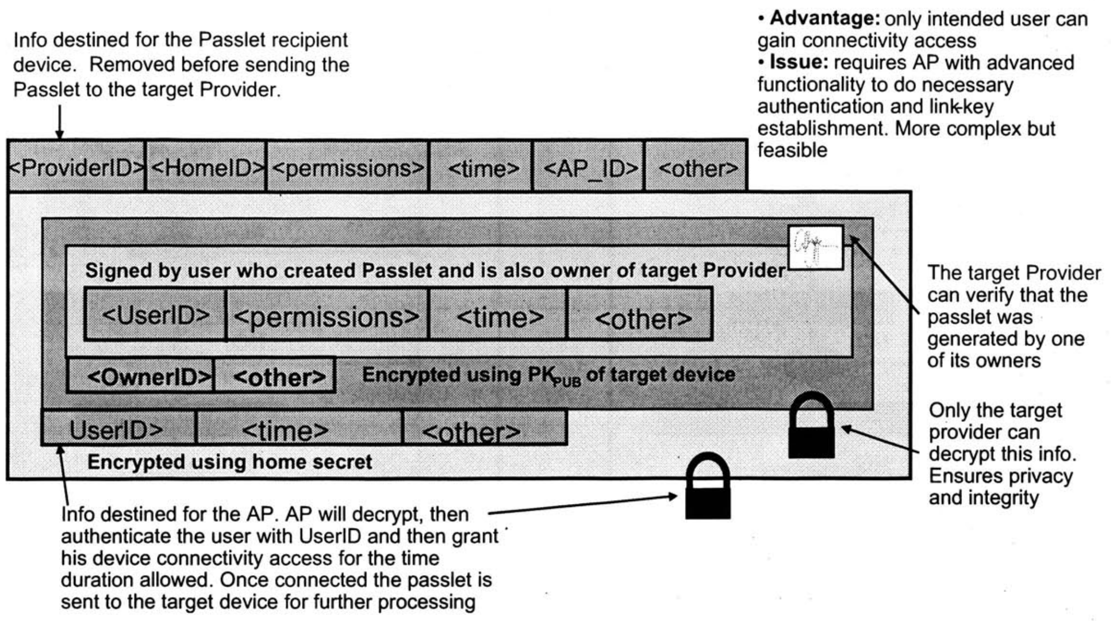

Info destined for the Passlet recipient device. Removed before sending the Passlet to the target Provider.
<ProviderID> <HomeID> <permissions> <time> <AP_ID> <other>
• Advantage: only intended user can gain connectivity access
• Issue: requires AP with advanced functionality to do necessary authentication and link-key establishment. More complex but feasible
Signed by user who created Passlet and is also owner of target Provider
<UserID> <permissions> <time> <other>
<OwnerID> <other> Encrypted using PK_PUB of target device
The target Provider can verify that the passlet was generated by one of its owners
UserID> <time> <other>
Encrypted using home secret
Only the target provider can decrypt this info. Ensures privacy and integrity
Info destined for the AP. AP will decrypt, then authenticate the user with UserID and then grant his device connectivity access for the time duration allowed. Once connected the passlet is sent to the target device for further processing

Figure 4-8: Illustration of the preferred Passlet design, which however requires a more intelligent access point than those currently available

mechanism than using public key cryptography for each message. Note further that this whole process is automated and requires no manual intervention on the part of the user.

The Passlet session establishment process requires mutual authentication between the Provider and the user. The Provider must authenticate the user in order to verify that s/he is in fact the bearer of the given Passlet. The user must also authenticate the Provider to ensure that s/he is not communicating with a malicious device. If this step were not present, then such a malicious device may pretend to be the expected Provider and through its interaction with the user obtain private information.

4.4.4 Passlet Revocation

Passlets can be manually revoked before their expiry time, but only by the creator of the Passlet and using the same device on which the Passlet was originally generated. The latter constraint is necessary since the user may own multiple devices and so may generate a Passlet from any of those devices. The other devices then would not know of this Passlet as there is no synchronization mechanism between all the devices.

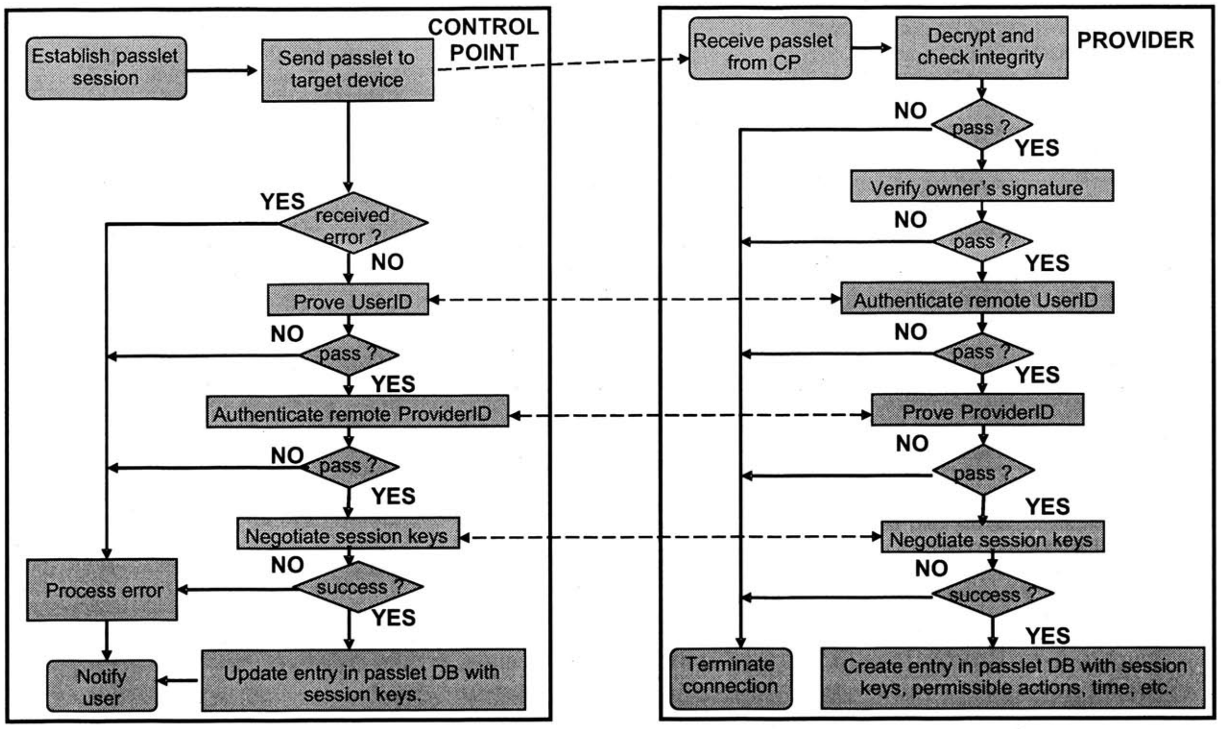


Figure 4-9: Establishment of a Passlet Session

The process by which revocation takes place is straightforward. The user simply browses the list of issued Passlets on the device and selects the one s/he wishes to revoke. A message that is digitally signed by the user is then sent to the IntuiSec software at the Provider indicating that the Passlet should be revoked. That user is by definition an owner of the Provider since otherwise s/he would not have been able to create the Passlet in the first place. The IntuiSec software at the Provider then adds the Passlet to a list of revoked Passlets. The bearer of that Passlet can then no longer access the Provider's functionality through use of that Passlet.

The revoked Passlets list is a non-decreasing list however its space requirements can be minimized through the use of a unique identifier for each Passlet. This identifier may be generated through straightforward mechanisms such as hashing the concatenation of the Passlet's contents with the current system time and a system hardware identifier of the device on which the Passlet is being generated. If this identifier were 16 bytes in length then a revocation list of 10,000 Passlets would result in a 160kB file. Since IntuiSec is to be used in a Smart Home environment, that number is far beyond what the system is most likely to handle and so the space requirements are feasible.

The length of the revoked Passlets list may also be reduced through the use of a global maximum for the Passlet duration. So for example if the maximum validity were one year, then all Passlets that are older than one year old would be safe to remove. IntuiSec could perform a weekly scan of the Passlet list in order to remove these expired Passlets.

4.5 User Interface - Security API

This part of the system calls for the creation of an API that enables user-level applications to leverage low level security mechanisms implicitly thereby allowing developers to focus instead on the functionality of the applications themselves. It also calls for the leveraging of these API's in order to present the user with a consistent and intuitive visualization of security-related aspects of the Smart Home. IntuiSec does not specify a particular graphical user interface (GUI) to be used; it leaves that to the application developers. One example of how such visualizations may be presented is given in Figure 4-10. In this example, the following properties are presented:

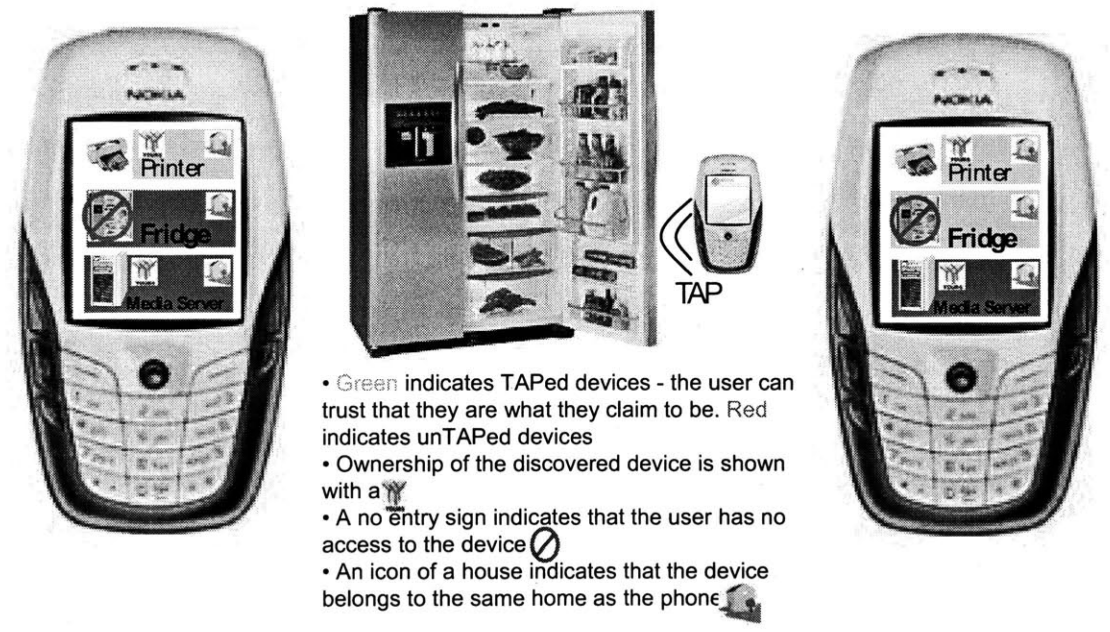
1. A house is used to show whether or not the device belongs to the same home as the user's device.

2. An icon is used to display whether or not the device is owned by the user.

3. A red background indicates an untrusted (i.e. unTAPed) device whereas a green background indicates a trusted (i.e. TAPed) device.

4. A no-entry sign informs the user that she does not have access to the displayed device.

It is important to note that the API is constructed such that the same calls may be made regardless of the underlying security infrastructure. This enables developers to create a consistent user experience over a variety of underlying security implementations.



• Green indicates TAPed devices - the user can trust that they are what they claim to be. Red indicates unTAPed devices
• Ownership of the discovered device is shown with a
• A no entry sign indicates that the user has no access to the device
• An icon of a house indicates that the device belongs to the same home as the phone

Figure 4-10: Security visualizations before and after establishing trust with the fridge

Chapter 5

Implementation

This chapter describes my implementation of the IntuiSec framework within a simulated Smart Home environment. The chapter begins within an overview of the implementation followed by a detailed description of each component of the software on the various devices.

5.1 Implementation Overview

This section gives an overview of the complete software architecture and the protocols used in this implementation. These components are detailed in Sections 5.2 – 5.5.

5.1.1 Implementation Functionality and Limitations

The implementation described here provides functionality for all three IntuiSec device types, namely Mobile, Fixed, and Infrastructure. Mobile Device functionality is implemented on the Nokia Series 60 2nd Edition Version 2.0 platform. Fixed and Infrastructure Devices are both implemented on the Linux platform. The implementation leverages UPnP as the underlying distributed services middleware with existing third party implementations of UPnP Provider software.

The following functionality is provided in this implementation:

Easy Setup – a process to take ownership of new devices and bootstrap them into the home network.

Building Trust – a process by which users can intuitively establish trusted relationships between devices. This enables the user to securely authenticate and reliably verify the identity of a remote device.

Granting Access with Passlets – a process by which users can easily grant access to both home occupants and visitors to devices that they own.

Security Visualization – this consists of a set of API's exposed by the framework to enable other applications to display security-related parameters in visualizations meaningful to non-expert users

There are some general points to note about this implementation:

- *Mobile Devices are the only Control Points.* UPnP Control point functionality is provided by a third party application on the phones and to demonstrate IntuiSec it is not necessary to have Control Point functionality on all the devices.

- *Public Key cryptography is not used.* The necessary cryptographic libraries are not part of the API's on the Series 60 cellular phones. However, IntuiSec uses well-established underlying cryptographic protocols and so this is a limitation of this particular implementation but not of the framework. A side effect of this is that the Passlets in this implementation are sent as clear text. Again this does not affect the demonstrability of IntuiSec.

- *The UPnP Providers are black-boxes requiring a workaround to provide access control.* Normally, a distributed system with a request/response framework would leverage the API's exposed by IntuiSec in order to provide access control. However, in the case of this implementation, UPnP is being retro-fitted with IntuiSec without the ability to make modifications to the UPnP software. To address this issue, this implementation of IntuiSec contains a workaround described in Section 5.1.5

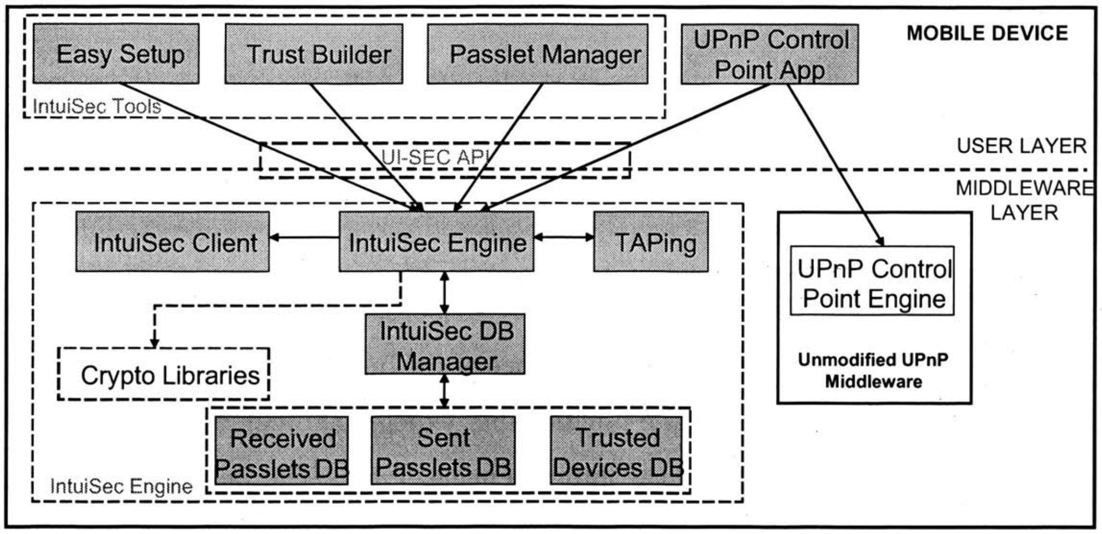


Figure 5-1: IntuiSec Mobile Device architecture on the Series 60 platform

5.1.2 Mobile Device Architecture

The IntuiSec Mobile Device software is written for the Nokia Series 60 2nd Edition Version 2.0 platform [49]. It is written in Symbian C++, which is the embedded environment that runs on Series 60 cellular phones. The main components of the software are the following, which are also depicted in Figure 5-1:

Easy Setup: A GUI application used to perform the three steps of the Easy Setup process. This module is discussed in more detail in Section 5.2

Trust Builder: A GUI application used to view already trusted devices and to create new trusted devices by TAPing. This module is discussed in more detail in Section 5.3

Passlet Manager: A GUI application used to view Passlets that have been sent, Passlets that have been received, and to create new Passlets. This module is discussed in more detail in Section 5.4.

UPnP Control Point: Although not part of IntuiSec itself, this module a modified UPnP Control Point application that makes use of the IntuiSec API. This application is used to interface with the UPnP Providers. This module is discussed in more detail in Section 5.5.

IntuiSec Engine: This module exports the IntuiSec API, which is used by the preceding GUI modules in order to access IntuiSec security functionality. It implements the layer of indirection between the system-level security mechanisms and the user-level concepts. The various classes that are part of the engine are all packaged into a DLL file called IntuiSecEngine.dll. When any of the user-level applications start, they create an instance of the IntuiSec Engine. This provides them with access to all of the functionality exposed through the IntuiSec API, which is exported through the IntuiSecEngine.dll file. The full API is given in Appendix A.

TAPing: This module handles communications over the Infrared channel and implements the TAPing Protocol, which is given in Appendix B. TAPing is implemented over infrared and follows a client/server model. The TAPing module on Mobile devices has both client and server functionality.

IntuiSec Client: This module handles the client portion of the IntuiSec Client and Server (ISCS) communications. It communicates over TCP/IP with the IntuiSec Server module located on Fixed Devices. The ISCS protocol is given in full in Appendix C.

IntuiSec DB Manager: This module is responsible for managing the various databases on the Mobile Device. The databases store information regarding received Passlets, sent Passlets, and trusted Devices.

5.1.3 Fixed and Infrastructure Device Architecture

The implementation for Fixed and Infrastructure Devices is done on the Linux platform. It is written in Java, however the code that handles the infrared communication is wrapped around a Java Native Interface (JNI) that is written for native Linux. Given the appropriate JNI's, it should be possible to run this part of the implementation on any platform that supports a Java Virtual Machine (JVM), however other platforms were not tested. The same code runs on both Infrastructure and Fixed devices; the software decides at runtime whether it is an Infrastructure Device or a Fixed Device by reading from a text-based configuration file, an example of which is given in Appendix D.

The main components of the software are described below and also shown in Figure 5-2. A subset of these components are active when the the device is an Infrastructure Device as opposed to a Fixed Device and this is shown in Figure 5-3.

Easy Setup: This is a GUI for the Easy Setup process, which presents the user with a series of dialogs to guide her through the process. The Easy Setup module implementation for Fixed Devices is similar to that of the Infrastructure Devices. The only difference is that in the third step of Easy Setup for a Fixed Device, the software displays a dialog to the user informing her that she needs to TAP the device with an already bootstrapped Mobile Device (to obtain the necessary connectivity-level security parameters). This contrasts with Infrastructure devices which generate their own sequence of keys that are then distributed to other Home Devices by TAPing (i.e. no third step is necessary). This process of key generation is hidden from the user.

System Tray Icon: This is a small icon that resides in the system tray and can be used to switch the TAPing module on or off.

Engine: This is the main component of the software. It handles the instantiation of and the communication between the other modules. It also configures the device appropriately based on startup configuration files and handles the progression through the Easy Setup process.

IntuiSec Server: This module implements the IntuiSec TCP/IP Server that handles the communications over the network with the IntuiSec TCP/IP Clients located on mobile devices that act as Control Points.

TAPing: Handles communications over the infrared channel. The TAPing module is implemented as a Java wrapper around a JNI to the Linux IrDA stack. It

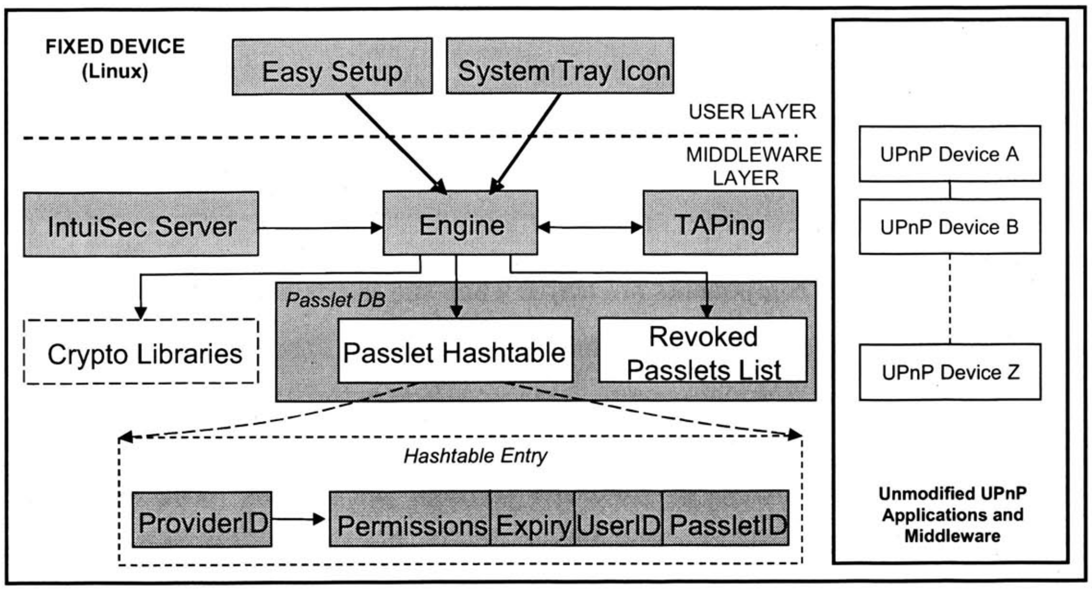


Figure 5-2: IntuiSec Fixed Device architecture on the Linux platform

consists of both an IrDA client and an IrDA server.

Passlet DB: This is a store for both valid and revoked Passlets. Valid Passlets are stored in a hashtable indexed by each Passlet's ProviderID. This done because a single IntuiSec device may run more than one UPnP provider, such as the laptop named COPERNICUS shown in the experimental setup in Figure 5-6. Revoked Passlets are stored as a list of PassletID's.

5.1.4 TAPing

TAPing is an intuitive user interaction modality that IntuiSec uses to exchange 2-way messages between devices when the user brings them in close proximity (i.e. the user "touches" the devices). It is used during Easy Setup, when building trust, and when handing out Passlets. TAPing is currently implemented using a Client/Server protocol over Infrared. All devices have a TAP server running on them that listens for incoming TAPs.

The first step in any TAP is for the devices to exchange their respective HomeIDs. There is no command for this - the TAPing client sends its HomeID immediately after

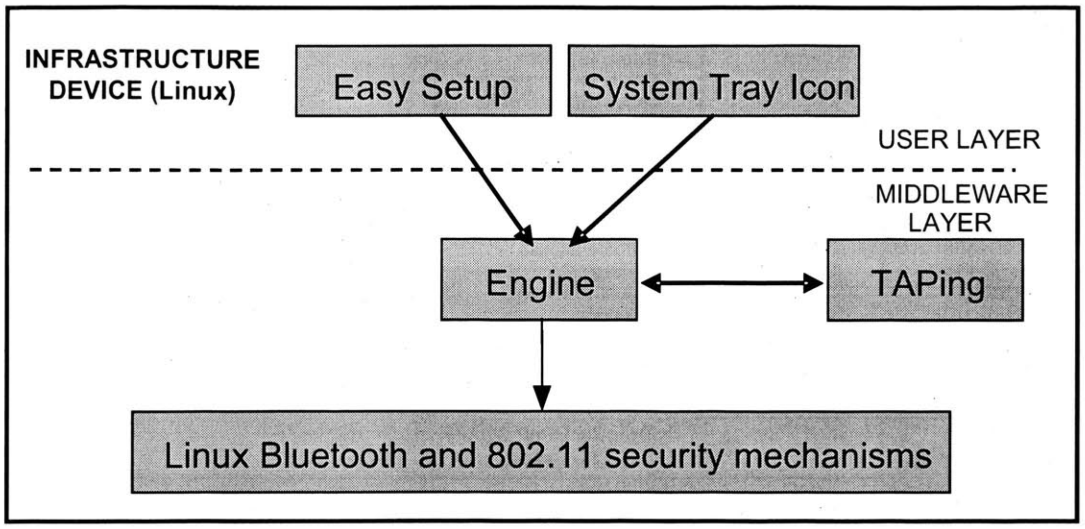

INFRASTRUCTURE DEVICE (Linux)
Easy Setup
System Tray Icon
USER LAYER
MIDDLEWARE LAYER
Engine
TAPing
Linux Bluetooth and 802.11 security mechanisms

Figure 5-3: IntuiSec Infrastructure Device architecture on the Linux platform

connecting with the TAPing server. The server then responds with its HomeID. If they do not match then the transaction enters 'unprivileged mode', where certain requests by the client are not honored by the server. In a real-word implementation this process would not involve a simple exchange of HomeID's, but rather a challenge-response protocol such as that prescribed by the design in Section 4.2. A complete list of the TAP protocol commands is given in Appendix B.

5.1.5 IntuiSec Clients and Servers (ISCS)

This part of the implementation is necessary only because the UPnP components are black boxes. In some cases the source code was not available, and in others it was too complex to modify for the purposes of demonstrating IntuiSec. Thus a workaround consisting of an IntuiSec Client on the Mobile Devices (Control Points) and an IntuiSec Server on the Fixed Devices (Providers) coordinate to provide the following functionality. This interaction takes place through the ISCS protocol given in Appendix C and it is also depicted in 5-4:

1. Establish Passlet sessions.

2. Revoke Passlets.

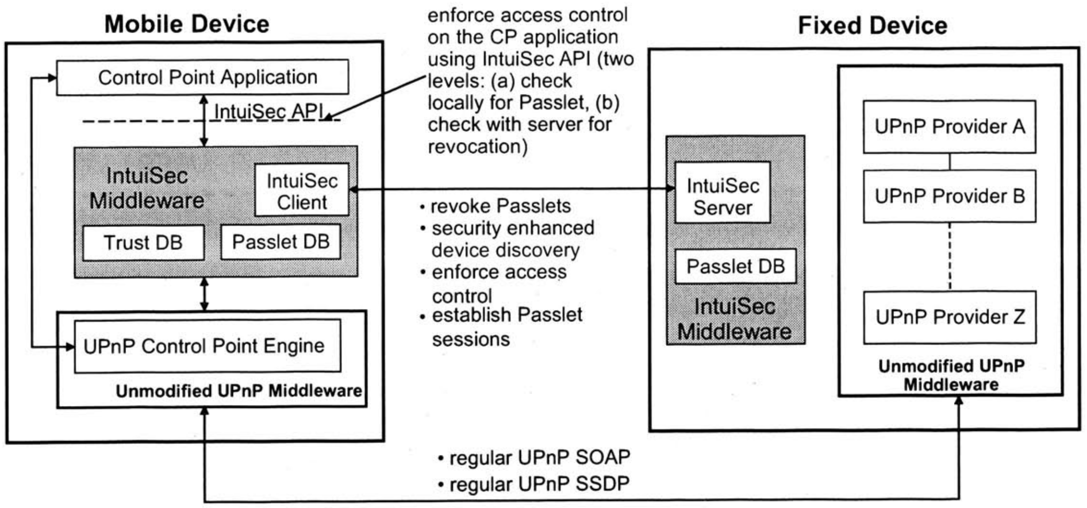

Mobile Device
Control Point Application
IntuiSec API
IntuiSec Middleware
IntuiSec Client
Trust DB
Passlet DB
UPnP Control Point Engine
Unmodified UPnP Middleware
enforce access control on the CP application using IntuiSec API (two levels: (a) check locally for Passlet, (b) check with server for revocation)
• revoke Passlets
• security enhanced device discovery
• enforce access control
• establish Passlet sessions
Fixed Device
IntuiSec Server
Passlet DB
IntuiSec Middleware
UPnP Provider A
UPnP Provider B
UPnP Provider Z
Unmodified UPnP Middleware
• regular UPnP SOAP
• regular UPnP SSDP

Figure 5-4: IntuiSec Client/Server mechanism enforces access control with existing UPnP and only a slight modification to the standard UPnP Control Point

3. Provide security enhanced device discovery enabling the display of meaningful security visualizations.

4. Enforce access control.

While this workaround emulates the user experience well, it does not enforce access control fully. Since the sending of UPnP actions is completely decoupled from IntuiSec, a standard UPnP Control Point completely bypasses this implementation of the Intuisec access control mechanism. However, the source code for a UPnP Control Point on the Series 60 mobile platform was available and the IntuiSec API is abstracted in such a way that it was straightforward to augment this Control Point and make it IntuiSec-enabled. This modification is described in more detail in Section 5.5. Note that this workaround is meant to be a temporary solution that allows this particular implementation to emulate the user experience and is not a part of the IntuiSec design.

The ideal solution is depicted in Figure 5-5 where the UPnP protocol is in fact augmented through its Simple Service Discovery Protocol (SSDP) and its Simple Object Access Protocol (SOAP) in order to provide the security enhanced device discovery and access control. That way no Control Point can bypass the IntuiSec

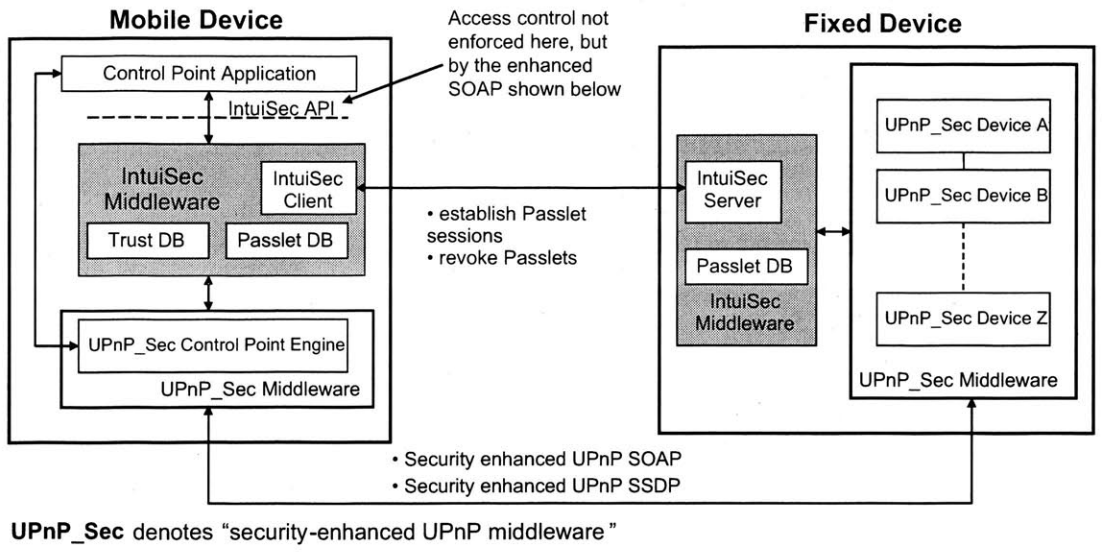

Mobile Device
Control Point Application
IntuiSec API
IntuiSec Middleware
IntuiSec Client
Trust DB
Passlet DB
UPnP_Sec Control Point Engine
UPnP_Sec Middleware
Access control not enforced here, but by the enhanced SOAP shown below
• establish Passlet sessions
• revoke Passlets
Fixed Device
IntuiSec Server
Passlet DB
IntuiSec Middleware
UPnP_Sec Device A
UPnP_Sec Device B
UPnP_Sec Device Z
UPnP_Sec Middleware
• Security enhanced UPnP SOAP
• Security enhanced UPnP SSDP
UPnP_Sec denotes "security-enhanced UPnP middleware"

Figure 5-5: IntuiSec Client/Server mechanism integrates with modified UPnP software on both the Control Points and the Providers

access control mechanism since it is embedded in the UPnP protocols and handled by the Providers. Note that Passlets are still handled by the IntuiSec Client and Server.

5.1.6 Experimental Setup

The simulated Smart Home environment consists of a standard D-Link Ethernet switch, three IBM ThinkPad laptops, and two Nokia 6600 cellular phones. The layout of the Smart Home is shown in Figure 5-6.

All three laptops run Ubuntu Linux [50], which is a Debian based distribution [51]. The first laptop uses a D-Link Bluetooth adapter and acts as a Bluetooth access point for the cell phones to connect to the home network. It also acts as a bridge between the wired segment and the Bluetooth segment. The second laptop acts as both a simulated Media Renderer and a Media Server, whilst the third laptop acts as a simulated Printer. Both these laptops run software implementations of UPnP devices created using the free Intel Authoring Tools from [52]. The cell phones both run the Nokia Series 60 2nd Edition Version 2.0 operating system [49], which is based on Symbian OS 7.0s [53]. All the devices except the Ethernet switch have Infrared connectivity. The various roles and owners of the devices are summarized in Table

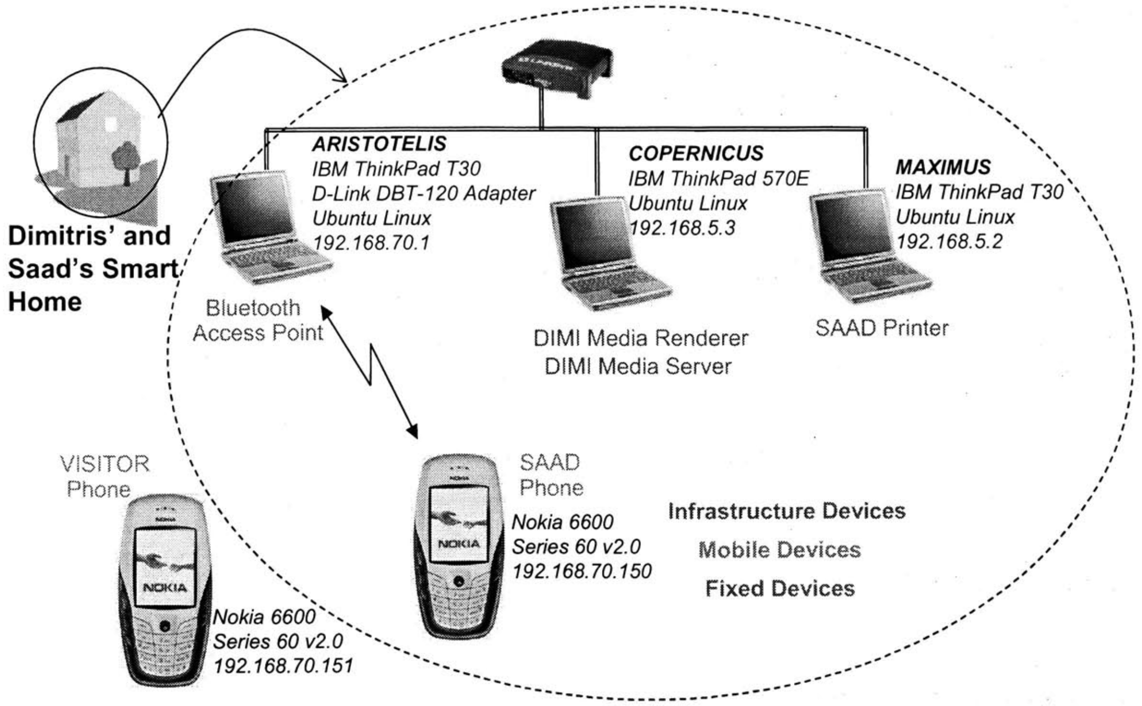


Figure 5-6: IntuiSec implementation experimental setup

5.1.

5.2 Easy Setup

The implementation allows for the bootstrapping of Mobile, Fixed, and Infrastructure devices into the Smart Home. After the Easy Setup process, a device is able to connect securely over Bluetooth or 802.11 without the need for the user to deal with Bluetooth PINs or WEP keys. The user interaction sequence is described next followed by an explanation of the application control flow during the process.

5.2.1 User Interaction View

The Easy Setup process on Mobile Devices consists of three ordered steps that are performed by the user. Screenshots of the Easy Setup application running on the Nokia 6600 mobile phones are given in Figure 5-7:

Device Name	IntuiSec Device Type	UPnP Device Type	Owner
ARISTOTELIS (Laptop)	Infrastructure	N/A	DIMI
COPERNICUS (Laptop)	Fixed	Providers (Media Server + Media Renderer)	DIMI
MAXIMUS (Laptop)	Fixed	Provider (Printer)	SAAD
SAAD Phone	Mobile	Control Point	SAAD
VISITOR Phone	Mobile	Control Point	VISITOR
Ethernet Switch	N/A	N/A	N/A

Table 5.1: Experimental setup – device roles and owners

Step 1: Input ownership information in the form of a username and password.

Step 2: Assign the home ID through insertion of the PhyKey - currently an MMC card as this is what the Nokia 6600 phones support.

Step 3: Retrieve connectivity information by TAPing an already set up Infrastructure Device.

The Easy Setup process on Fixed Devices also consists of three ordered steps that are performed by the user. Screenshots of the Easy Setup application running on the Linux Laptops are given in Figure 5-8:

Step 1: Input ownership information in the form of a username and password.

Step 2: Assign the home ID through insertion of the PhyKey - currently an MMC card through a USB MMC reader to maintain consistency with the process on the Nokia cell phones.

Step 3: Retrieve connectivity information by using an already set up Mobile Device to TAP the Fixed Device.

The Easy Setup process on Infrastructure Devices is identical to that of Fixed Devices except that the third step is not needed since the Infrastructure Device generates its own sequence of keys that are then handed out to other Home Devices when they are set up.

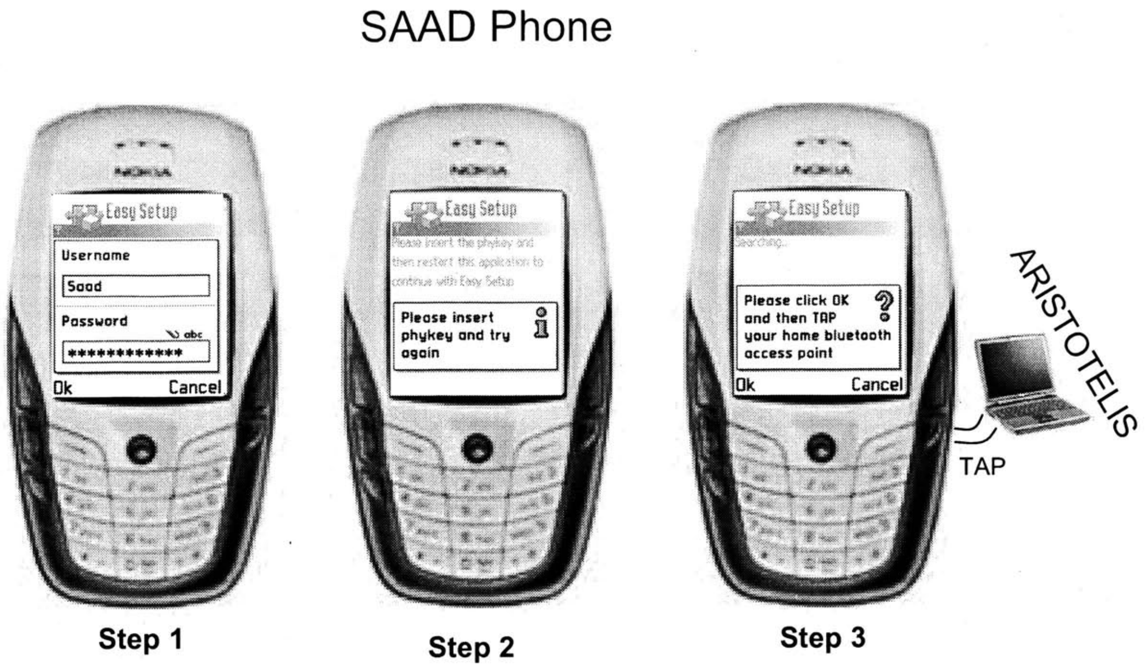
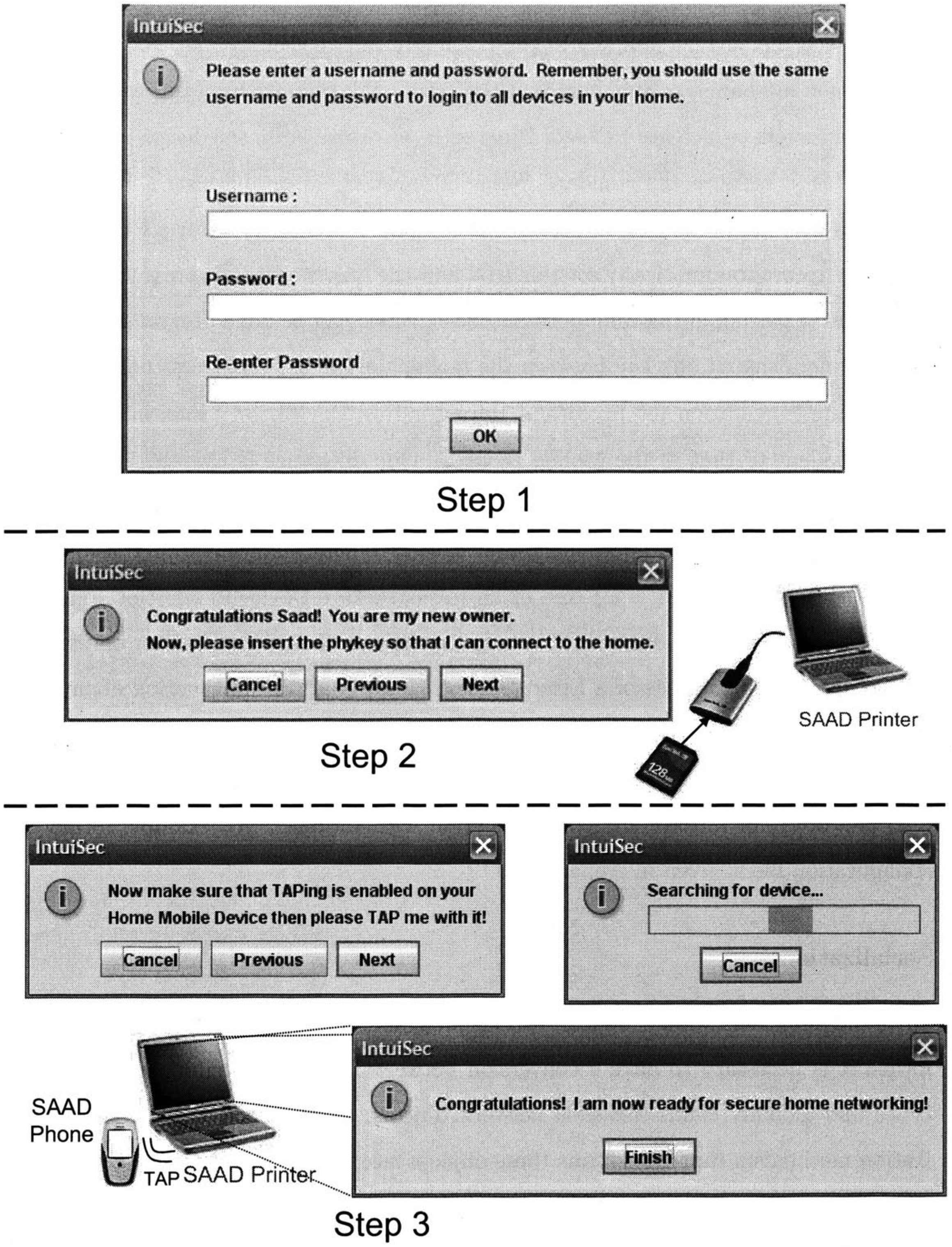

SAAD Phone
Easy Setup
Username
Saad
Password
abc
****
Ok
Cancel
Easy Setup
Please insert the phykey and then restart this application to continue with Easy Setup
Please insert phykey and try again
Easy Setup
Searching...
Please click OK and then TAP your home bluetooth access point
Ok
Cancel
ARISTOTELIS
TAP
Step 1
Step 2
Step 3

Figure 5-7: User's view of performing Easy Setup on a Mobile Device - the Nokia 6600 cell phone

5.2.2 System View

On the Mobile Devices, the Easy Setup application uses the first six functions of the IntuiSec API given in Appendix A. The application control flow is described by the activity diagram in Figure 5-9. The UML sequence diagrams in Figures 5-10 and 5-11 show the TAPing portion (Step 3) in more detail highlighting the difference between an attempt by a Home Device to perform Easy Setup and a Visitor Device. Note that information obtained during each step of the process is stored in files. This includes the OwnerID, HomeID, and the list of connectivity keys. The middleware probes for the existence of these files to determine which step of the process it is at in case the user does not complete the whole process at one sitting. In Step 3 of the process after the connectivity information is received from the access point, the middleware automatically creates a Bluetooth pairing in the pairing database on the Mobile Device and inserts the first of the list of Bluetooth keys as the current Bluetooth Link Key. This enables the Mobile Device to connect securely to the access point without any further configuration from the user. The rest of the keys are stored


IntuiSec

Please enter a username and password. Remember, you should use the same username and password to login to all devices in your home.

Username :

Password :

Re-enter Password

IntuiSec

Congratulations Saad! You are my new owner.
Now, please insert the phykey so that I can connect to the home.

IntuiSec

Now make sure that TAPing is enabled on your Home Mobile Device then please TAP me with it!

IntuiSec

Searching for device...

IntuiSec

Congratulations! I am now ready for secure home networking!

Figure 5-8: User's view of performing Easy Setup on a Fixed Device - the IBM laptop acting as a printer

in a file. There is a function that extracts the next 16-byte key from the file and uses it to replace the current link key in the pairing database thereby implementing the key-rotation mechanism. Note here that the actual link keys are used rather than the user-level secrets (which are PINs for Bluetooth) as described in the design in Section 4.2.1. This is because the standard Series 60 method for establishing a Bluetooth pairing is through presenting the user with a popup to manually enter a PIN. There is no way to programmatically set this PIN and the only way to suppress the popup is to create a pairing in the Bluetooth database, which can be done programmatically by inserting a shared link key between the mobile device and the access point.

The implementation for Fixed and Infrastructure Devices on Linux is functionally equivalent to that of the Mobile Devices. One difference is that on Fixed and Infrastructure Devices, IntuiSec uses a configuration file upon initialization. This configuration file contains information that is used to determine the properties of the device, such as whether it is a Fixed or an Infrastructure Device in addition to data specific to each type of device. This way the same software can be run on either type of device. In the case of a Fixed Device the file contains information about all the UPnP Providers on that device. This is necessary for IntuiSec to become aware of the UPnP Providers on that device since the UPnP Provider software is unmodified and does not initiate any communication with IntuiSec. An example IntuiSec configuration file is given in Appendix D.

Serialization

Since the implementation is done in various programming languages on different platforms, it is necessary to have a conversion between language-specific abstract data types and globally comprehensible network objects. This occurs through a serialization mechanism that transforms these objects into the form [<length1> <field1> <length2> <field2>... <lengthN> <fieldN>] where <length1>... <lengthN> are 4-byte integers representing the total length of the corresponding fields <field1>... <fieldN> that follow. One example of such a serialized object is the `serialized_conn_info` object depicted as part of the sequence diagrams in Figure 5-10. Besides connectivity

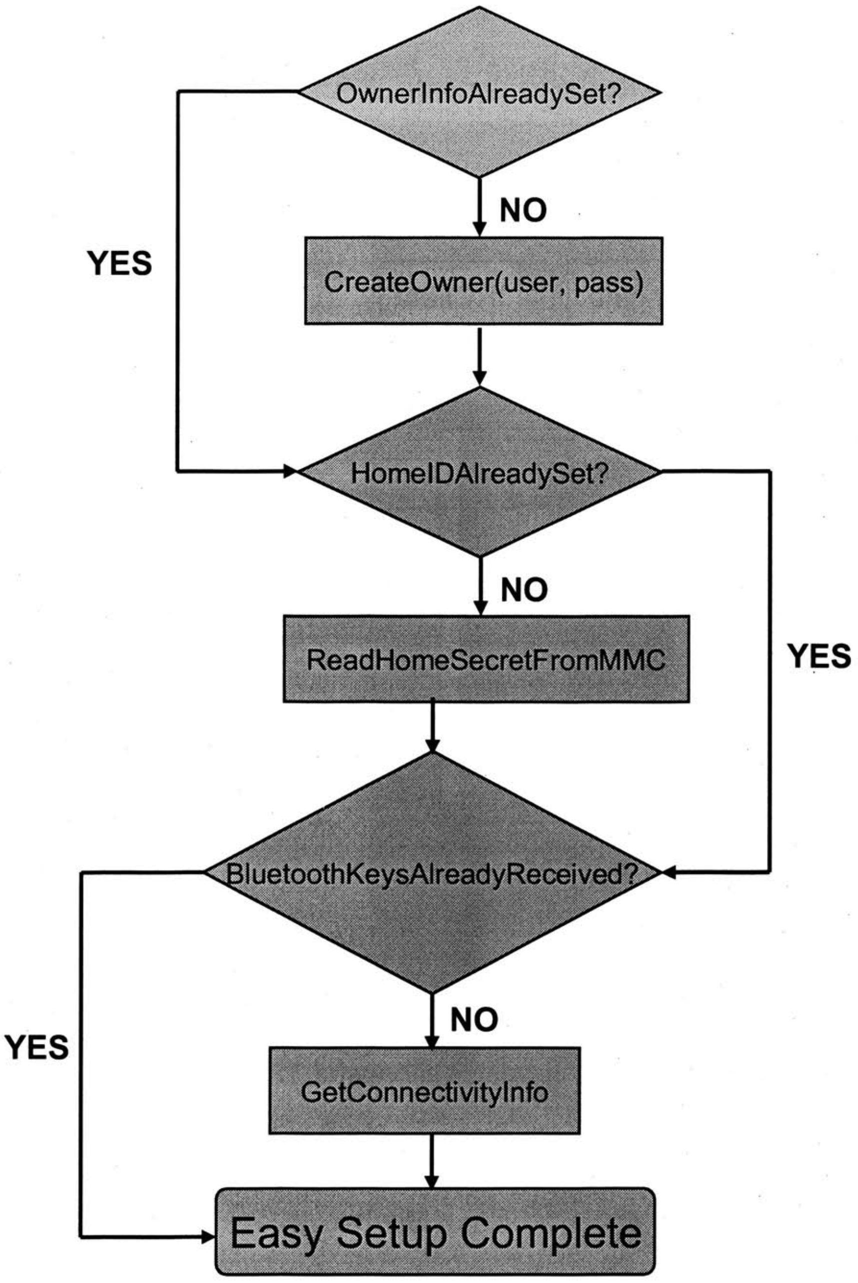


Figure 5-9: Activity Diagram of the Easy Setup process on Series 60 phones

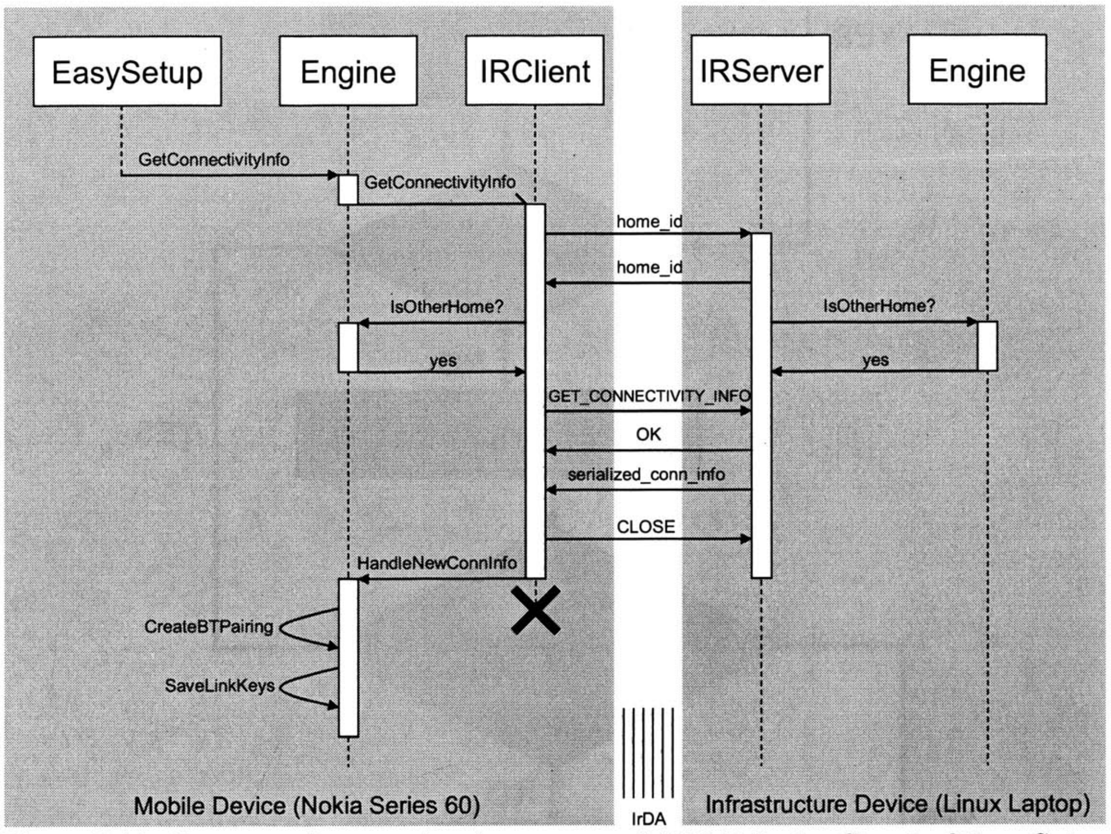


Figure 5-10: Sequence diagram showing a successful TAP during Step 3 of Easy Setup process

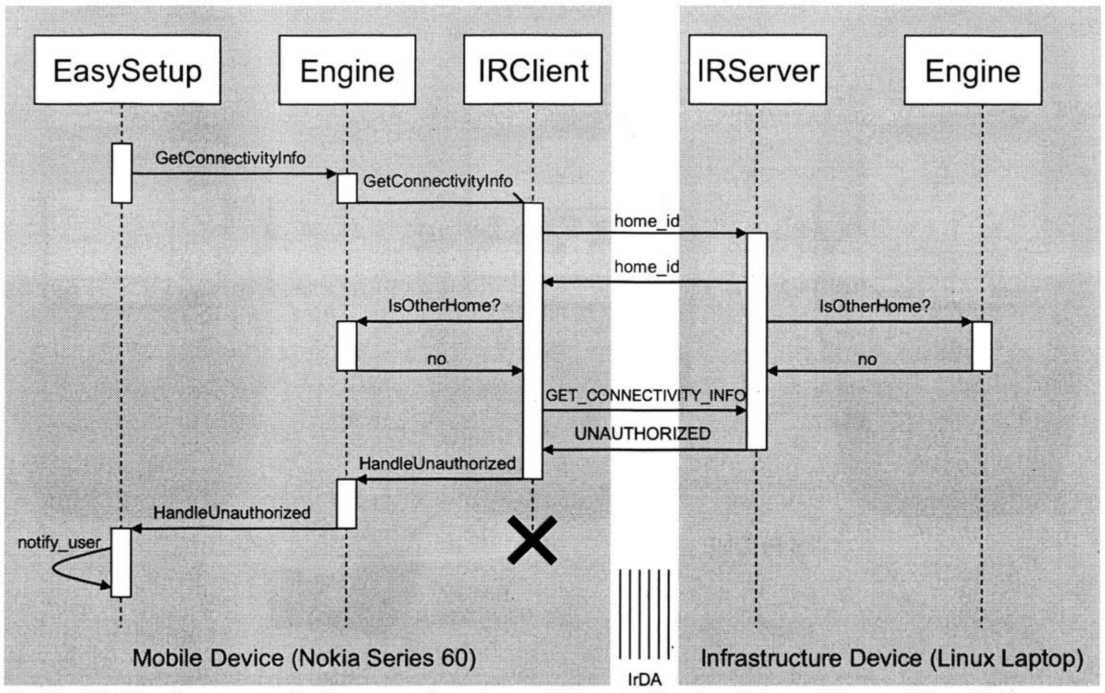


Figure 5-11: Sequence Diagram showing an unauthorized attempt at obtaining connectivity information during Step 3 of the Easy Setup process

information, the other abstract data types are the trusted device information and Passlets. The process of serialization for all data types is identical. The three data types and their internal attributes are listed in Appendix E

5.3 Trust Builder

5.3.1 User Interaction View

The user interaction process with the Trust Builder application is straightforward. The user selects the "Build Trust" option after which she is asked to TAP the device she wishes to build trust with. Once the TAP is complete, the device appears in the user's trusted devices list. This process is shown in Figure 5-12

5.3.2 System View

The Trust Builder Series 60 application uses `GetTrustedDevices()` to obtain a list of trusted devices and uses that list to populate a display of those devices. The

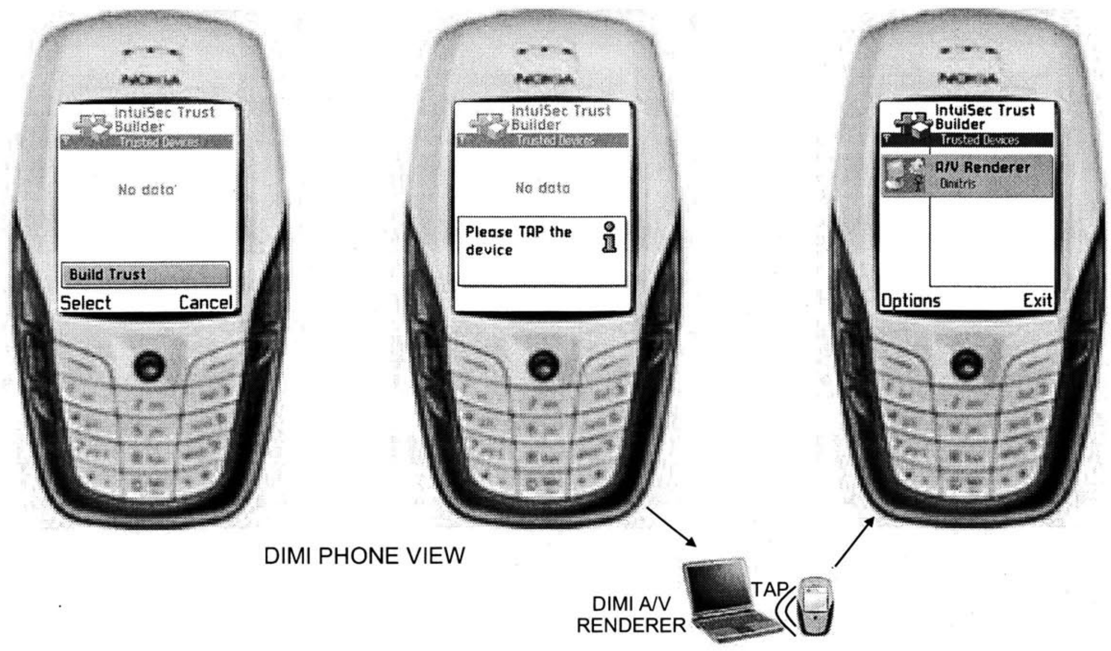


Figure 5-12: User interaction with the Trust Builder application on the Nokia 6600

user may opt to delete a device from the database, in which case the application calls `DeleteTrustedDevice()`. If a user wishes to build trust with a new device, she selects "Build Trust" from the options menu and then she is asked to TAP the device she wishes to build trust with. Analogous to the Easy Setup retrieval of connectivity information, the Engine creates an instance of the IRClient and calls `GetDeviceInfo()` . The IRClient then proceeds to retrieve the device information by using the `GET_DEVICE_INFO` command of the TAPing protocol. The IRServer on the side of the Fixed Device then responds with the device information of every UPnP provider that is running on that device. It obtains this information from the configuration file described in Section 5.2.2. The complete process is illustrated by the sequence diagram in Figure 5-13

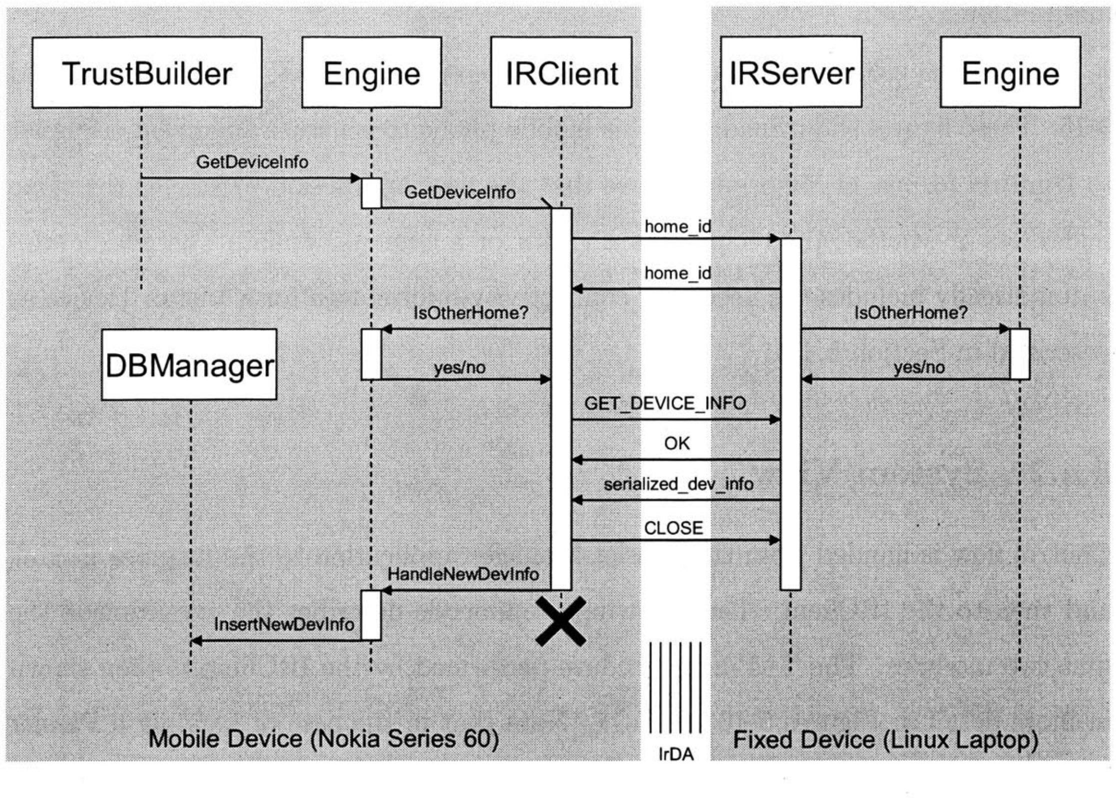


Figure 5-13: Sequence diagram showing the process of building trust

5.4 Passlet Manager

5.4.1 User Interaction View

The implementation of Passlets along with the Passlet Manager application creates a user experience identical to that which is described in Section 4.4. The Passlet Manager application contains three tabs. The first lists Passlets that the user has sent with some information about each, such as the device which it is for, the time until expiry, and the recipient. The second lists Passlets that the user has received with some information about each, such as the time until expiry, the device which it is for, and the user who generated the Passlet. The third tab displays devices that the user owns and has TAPed. These are the devices which the user is able to generate Passlets for. Since a core component of Passlets is the ability to incorporate trusted authentication information pertaining to the device which a Passlet is allowing access to, it is necessary for the user to first TAP the device in order to obtain that information.

The steps a user takes to create and send a passlet are shown in Figures 5-14 and 5-15. These figures show Saad using his mobile phone to generate and grant a Passlet to Dimitris for use of his printer. Note that the user interaction process is the same regardless of whether the recipient is a visitor or a home occupant. The middleware automatically includes the necessary connectivity information for a Visitor Device as described in Section 5.4.2.

5.4.2 System View

Control flow is handed from the Passlet Manager application to the IntuiSec Engine and then to the IRClient. The following pseudocode describes the execution of the first two modules. The TAPing procedure performed by the IRClient is then shown in more detail in Figures 5-16 and 5-17. Note that in the case of handing a Passlet to a visitor, the connectivity information is sent separately however this is merely to simplify the implementation. In the design, connectivity information is incorporated into the internals of each Passlet.

```
PassletManager::CreatePasslet {
    display_owned_devices;
    display_default_passlet_template_for(selected_device_type);
    passlet = Passlet.create_passlet_from_user_params;
    IntuiSecEngine.SendPasslet(passlet);
}

IntuiSecEngine::SendPasslet(passlet) {
    passlet.ownerID = this.get_owner_id();
    passlet.passletID = generate_passlet_id();
    IRClient.SendPasslet(passlet);
}
```

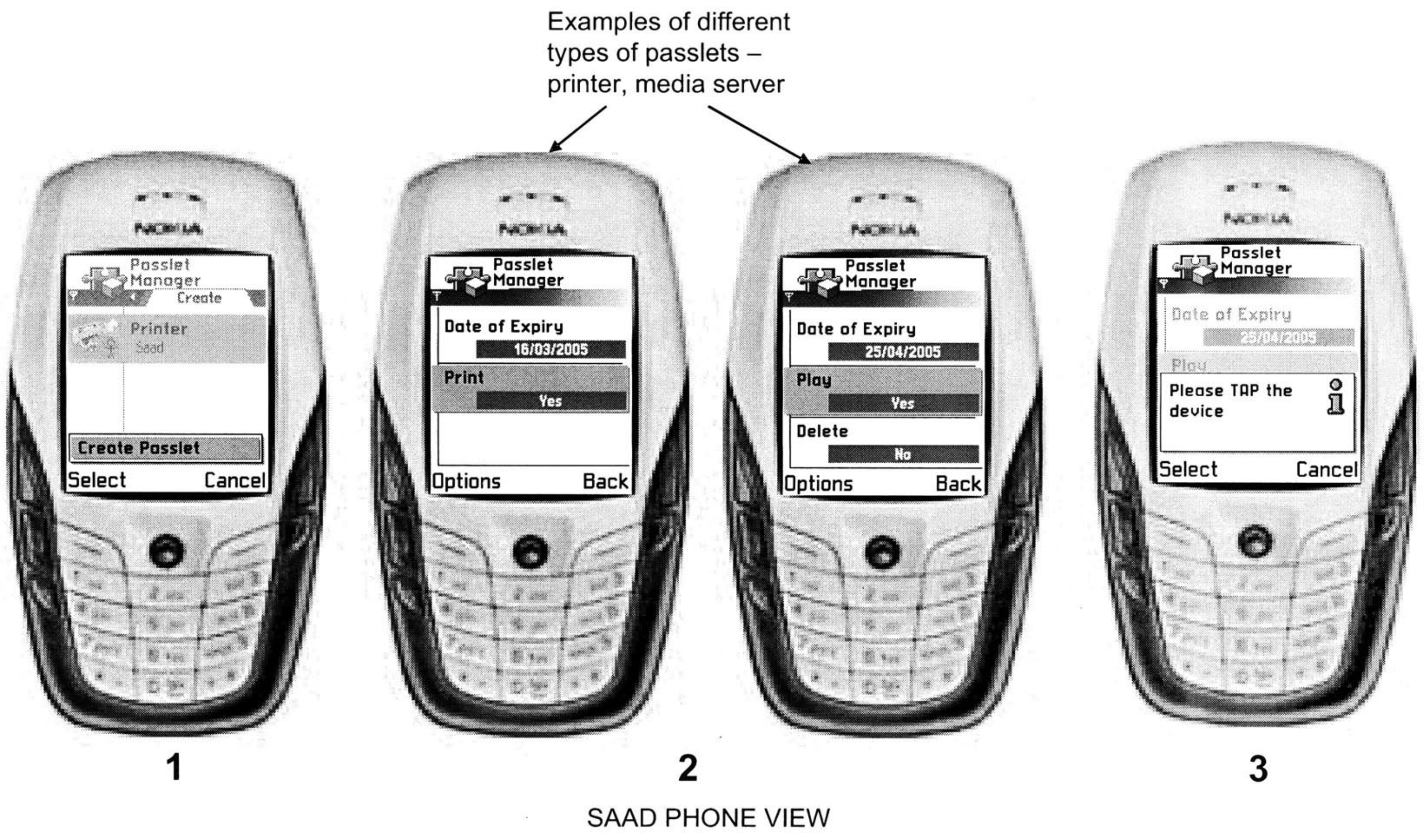


Figure 5-14: Saad using the Passlet Manager application on the Series 60 phones to create a Passlet for his printer

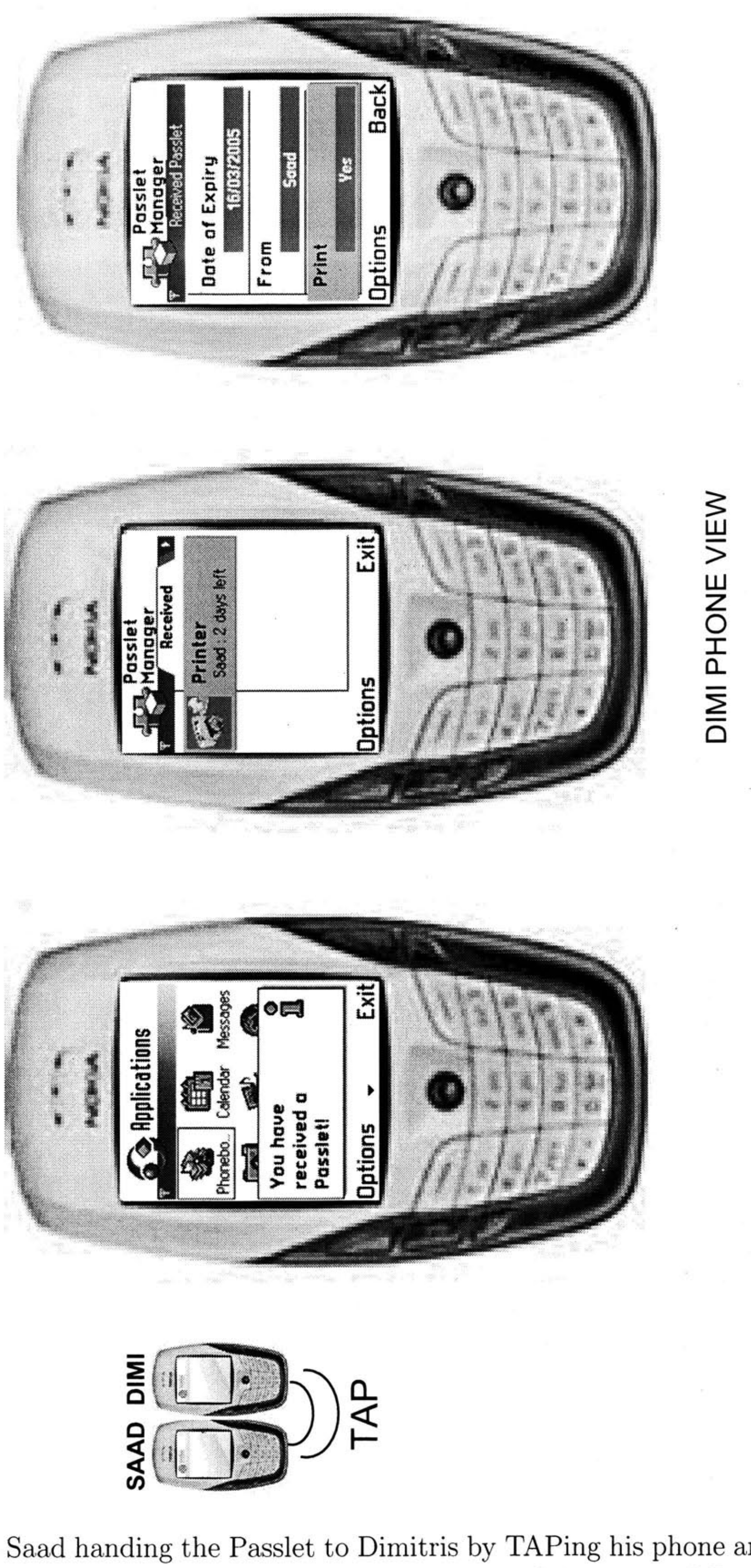


Figure 5-15: Saad handing the Passlet to Dimitris by TAPing his phone and the view on Dimitris' phone after the TAP showing the details of the received Passlet

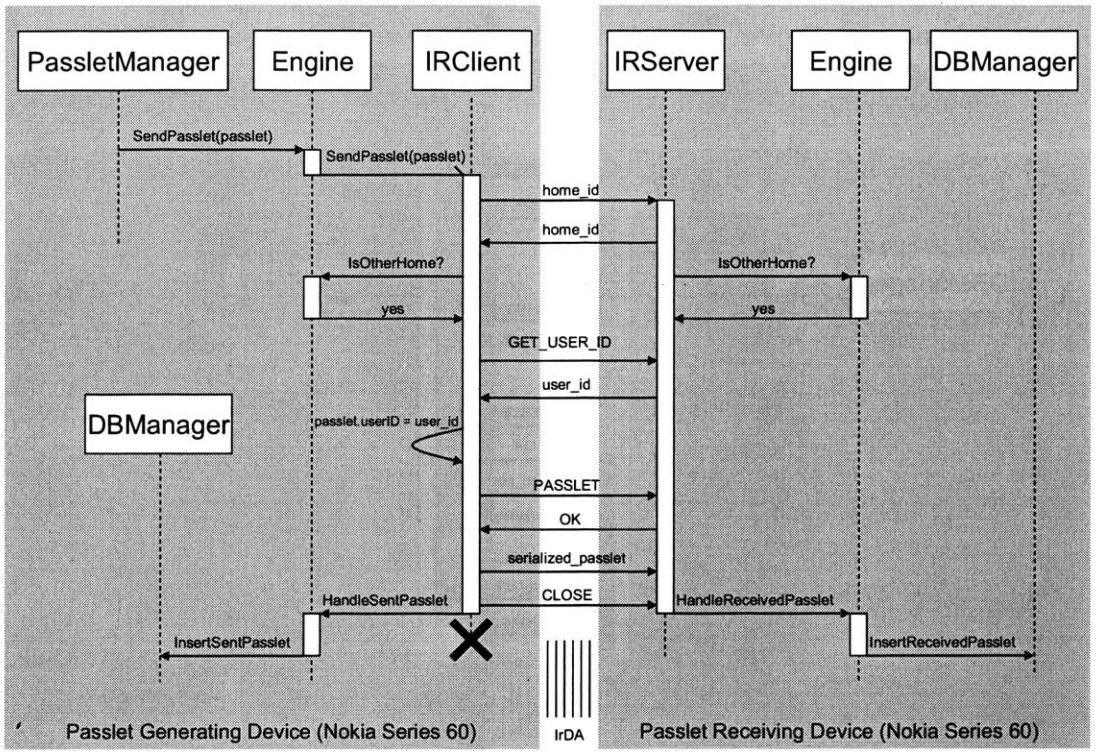


Figure 5-16: Sequence diagram showing the TAPing segment of sending a Passlet to a Home Device

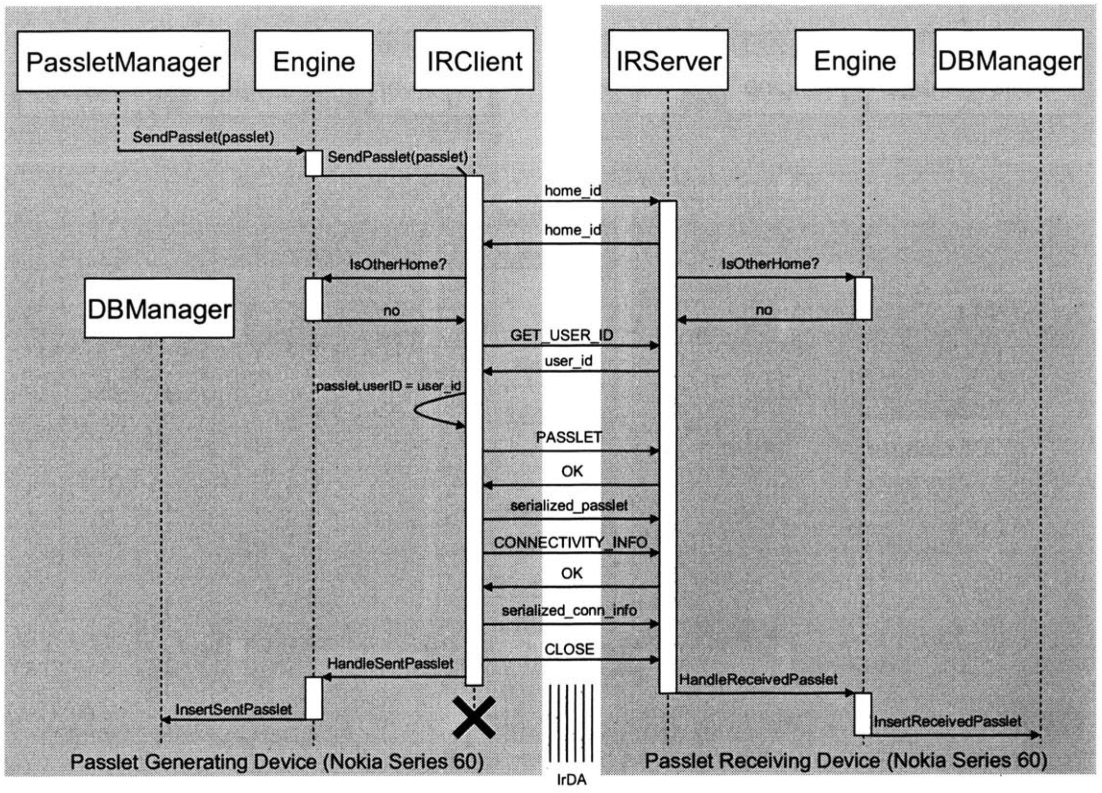


Figure 5-17: Sequence diagram showing the TAPing segment of sending a Passlet to a Visitor Device

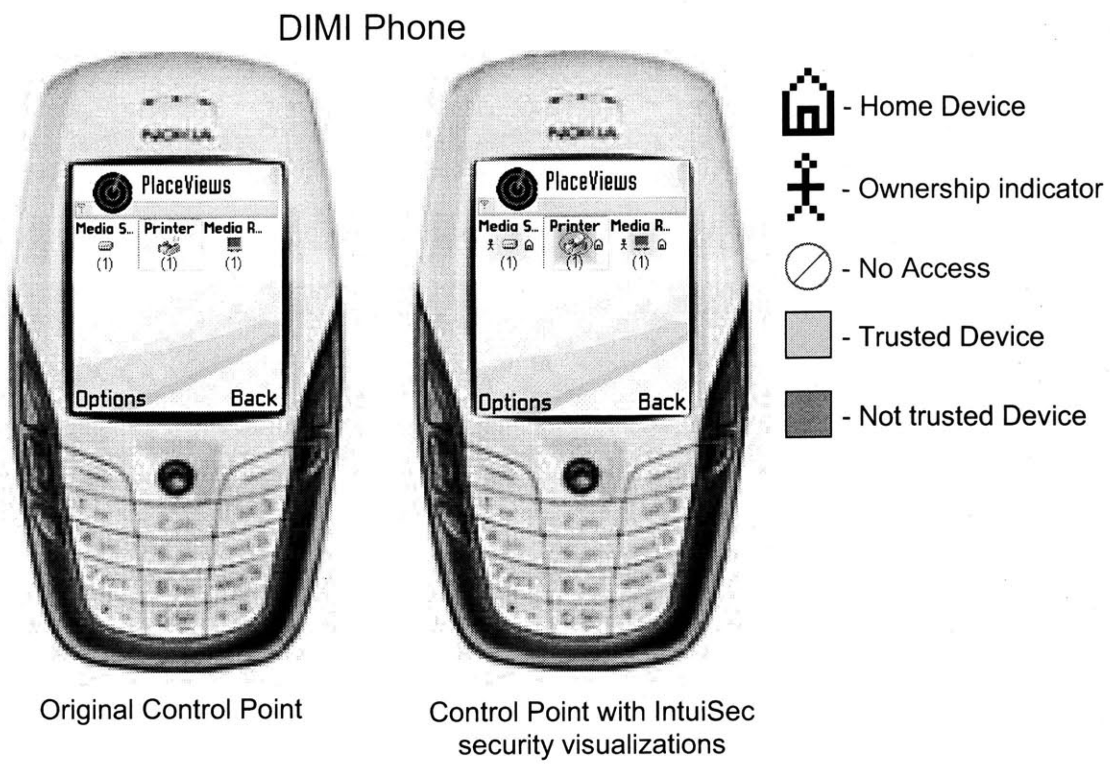


Figure 5-18: Application view of Dimitris using his mobile phone to browse the home network with and without security visualizations

5.5 IntuiSec-enabled Control Point

5.5.1 User Interaction View

After interacting with the system and the various devices in the home, a user is able to view the results of these interactions through certain security visualizations. These visualizations are displayed when a user explores the home network and browses for devices within the home. This implementation uses UPnP as the underlying device discovery and control framework. Figure 5-18 shows views of the Control Point application with and without the security visualizations.

5.5.2 System View

The IntuiSec-enabled Control Point application is augmented to provide both security enhanced device discovery and access control functionality.

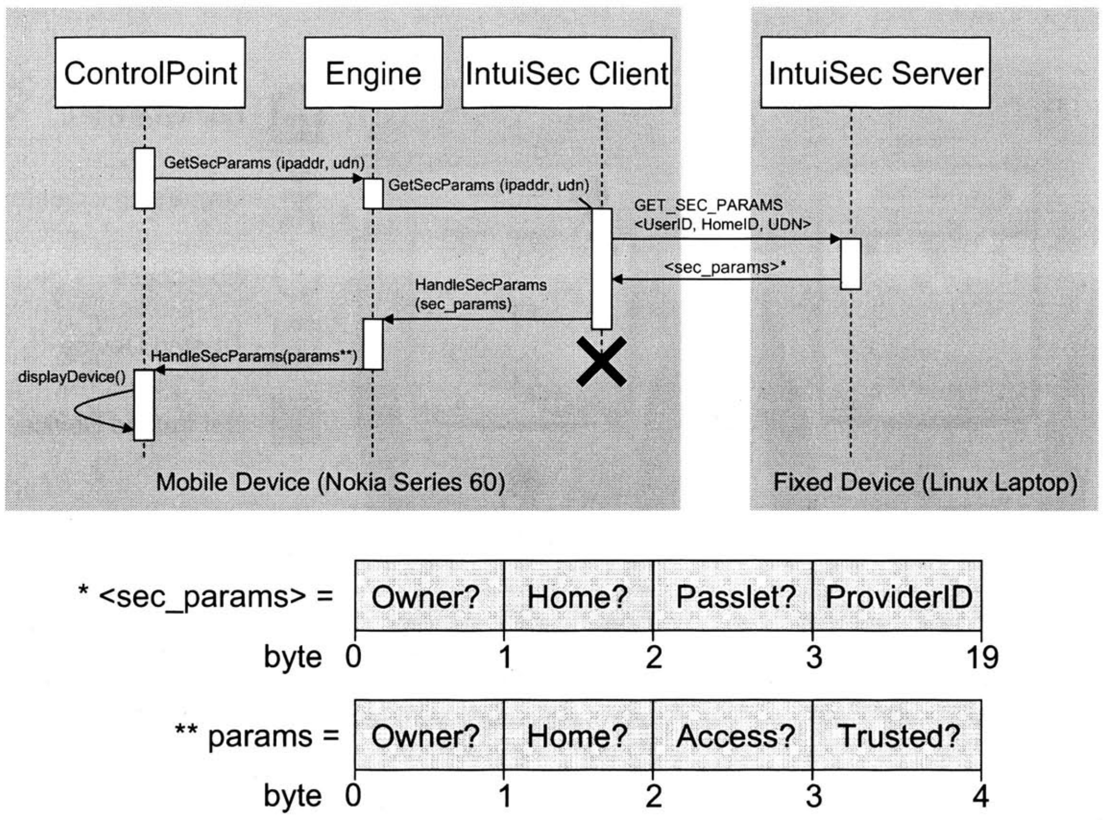

ControlPoint
Engine
IntuiSec Client
IntuiSec Server
GetSecParams (ipaddr, udn)
GetSecParams (ipaddr, udn)
GET_SEC_PARAMS <UserID, HomeID, UDN>
<sec_params>*
HandleSecParams (sec_params)
HandleSecParams(params**)
displayDevice()
Mobile Device (Nokia Series 60)
Fixed Device (Linux Laptop)
* <sec_params> = Owner? Home? Passlet? ProviderID
byte 0 1 2 3 19
** params = Owner? Home? Access? Trusted?
byte 0 1 2 3 4

Figure 5-19: Sequence diagram showing how the Control Point application obtains the security parameters from the various UPnP providers in order to display the relevant visualizations to the user

Security Enhanced Device Discovery

The IntuiSec-enabled Control Point application performs a regular UPnP discovery and for each response from a Provider that it receives it makes a separate API call to `GetSecParams()`. This process is shown in the sequence diagram in Figure 5-19. When this function completes it returns a 4-byte that indicates:

1. Whether or not the user making the request owns the device on which the discovered Provider resides. Recall that a single IntuiSec device may run multiple UPnP Providers, such as the laptop named COPERNICUS in the experimental setup described in Section 5.1.6 which has both a Media Renderer and a Media Server.

2. Whether or not the device making the request and the device on which the

discovered Provider resides both belong to the same home.

3. Whether or not the user making the request has access to the discovered Provider. This is determined either through the possession of a valid Passlet to or through ownership of the remote Provider.

4. Whether or not the discovered device is a trusted device - trust having been established through the Trust Builder application.

It is important to note that the IntuiSec software on the Fixed Device which hosts the Providers is responsible for translating between UPnP identification strings (known Unique Device Names or UDNs) and IntuiSec's ProviderIDs. The IntuiSec software on a Fixed Device becomes aware of the UDNs of all the Providers that are running on that Fixed Device through the IntuiSec configuration file described in Section 5.2.2 and given in Appendix D. Once the ProviderID is sent back to the IntuiSec Client making the request, the client can then proceed to check whether or not the user has a Passlet for the specified device and whether the device is trusted or not.

Access Control

After device discovery has been performed, the user normally selects the device s/he wishes to access. Upon selection, the Control Point application retrieves the UPnP device description from the remote Provider and displays the appropriate control view to the user. The available user-level actions in the control view vary depending on the control functionality of the device that is chosen. So a printer for example would display 'Print', 'Delete Job', etc. whereas a media server would display 'Upload', 'Download', etc. When the user selects an action to perform, the Control Point application automatically translates that user-level action into the necessary sequence of UPnP SOAP actions, which individually are not meaningful on a user level. Instead of automatically performing this translation however, the IntuiSec-enabled Control Point application first checks to see if the user has access to the given action through a call to the `GetActions()` function that is part of the IntuiSec API. This function retrieves a comma separated list of user-level actions that the user is allowed to perform on the Provider. If the user has access to the given action, then the regular UPnP SOAP communications ensue. Otherwise the user is notified that s/he does not have access.

The list of actions that is retrieved from the Provider will vary depending on the situation:

1. If the user is the owner of the Provider, then the list of actions will be a complete list of all available user-level actions because the owner by default has access to all the functionality on the Provider.

2. If the user is not the owner but has a Passlet for which there is an already established Passlet session with the Provider, then the list of actions will consist of all allowed actions specified in the Passlet.

3. If the user is not the owner and does not have a Passlet session established yet but is in possession of a Passlet, then the IntuiSec Client first posts the Passlet using the `PASSLET` command described in Appendix C. After posting the Passlet a session is established and communication proceeds as in case 2.

4. If the user is not the owner and does not have a Passlet to access the device or a valid Passlet session, then list of actions will be an empty list and the user will not have access to any of the functionality on the Provider.

Chapter 6

Conclusion and Future Directions

6.1 Discussion of Results

This thesis began with an illustration of the clear need to readdress the design of current security frameworks. I claimed that most current frameworks are inadequate not in their cryptographic correctness, but in their lack of focus on the users who are going to use these frameworks. As a solution to this problem I presented IntuiSec, which is a security framework for Smart Home environments with a unique focus on usability. I showed how this framework is intuitive and implicit in its functionality enabling average users to avoid getting caught up in the intricacies of cryptographic mechanisms. In the design I demonstrated how IntuiSec enables users to easily incorporate new devices securely into the Smart Home environment in three simple steps, how users can build trusted relationships between different devices by simply touching them, how they may grant access to other home occupants and visitors also using touch, and how they can intuitively visualize the security-related aspects of the Smart Home environment. Most importantly IntuiSec presents the user with this consistent interaction regardless of the underlying cryptographic mechanism, be it 802.11 with WEP or WPA, Bluetooth PINs, or a variety of other connectivity security implementations.

I later presented a working instantiation of IntuiSec using both the Nokia Series 60 and the Linux platforms within a simulated Smart Home environment. The implementation demonstrates both the efficaciousness and feasibility of the functionality described in the design of IntuiSec.

6.2 Future Directions

IntuiSec is clearly a large step forward in the design of security frameworks for complex environments that are meant to be managed by non-technical users. Nevertheless, there is still room for much research in this field. One interesting area would be to extend IntuiSec to situations where the user is outside their home environment but wishes to access functionality within the home. Remote access is becoming increasingly important because as homes provide more functionality en route to becoming Smart Homes, individuals are increasingly feeling the need to control and observe their homes from anywhere. This may be done through some form of Virtual Private Network (VPN) or secure tunneling. One challenge with VPN's is appropriately handling NAT traversal. Also current VPN's are difficult to configure and a new abstraction similar to those presented in this thesis could be used to create a more usable experience.

In the longer run it is also necessary to consider how users will interact with security frameworks outside of their homes - in their offices, in hotels, in airports, and other locations. In public spaces, usability is extremely important due to the lack of readily available technical support. The security requirements of public spaces are also different than those of Smart Homes since public spaces are more susceptible to physical attacks (both wired and wireless). In the enterprise, tech support is available however businesses would save a lot on their help desk expenses by having a more usable security framework in the office. They would also improve the security of sensitive corporate material and the productivity of their employees who would be able to focus entirely on getting their jobs done rather than dealing with intricate security measures that many times simply get in the way. It should be feasible to extract the consistent principles used by IntuiSec and apply them to the state of ubiquitous connectivity that we will experience in the near future.

Appendix A

IntuiSec Engine API

API Function Description	Used By
Checks to see if an owner has already been assigned to this device. `OwnerInfoAlreadySet? ()`	Easy Setup – Step 1
Creates a new owner for the device. `CreateOwner (username, password)`	
Checks to see if the device has already been assigned a homeID. `HomeIDAlreadySet? ()`	Easy Setup – Step 2
Reads the Home Secret from the MMC card and assigns it to this device. `AssignHomeSecretFromMMC ()`	
Checks to see if the device has already received connectivity information to connect to the home network. `BluetoothKeysAlreadyReceived? ()`	Easy Setup – Step 3
Gets connectivity information by TAPing and creates a Bluetooth pairing and the bluetoothkeys.bin file. `GetConnectivityInfo ()`	
Retrieves a list of trusted devices. `GetTrustedDevices ()`	Trust Builder
Gets Trusted Provider information through a TAP. `GetProviderInfo ()`	
Deletes the specified device from the Trusted Devices database. `DeleteTrustedDevice (deviceID)`	
Retrieves a list of the passlets the current user has received. `GetPassletsReceived ()`	Passlet Manager – Received Passlets
Retrieves a list of the passlets the current user has given out. `GetPassletsGiven ()`	Passlet Manager – Sent Passlets
Sends a revocation message for the specified passlet to the device at the specified address. `RevokePasslet (deviceID, passletID)`	
Retrieves a list of devices that the current user owns and has TAPed. `GetOwnedDevices ()`	Passlet Manager – Create Passlet
Sends the given passlet through a TAP and inserts it into the Sent Passlets database if successful. `SendPasslet (passlet)`	
Gets the security parameters for the device with the specified UDN at the specified IP address. `GetSecParams (IPAddress, UDN)`	UPnP Control Point Application
Gets the list of user-level actions that the user has permission to use on the given device. `GetActions (IPAddress, UDN)`	

Figure A-1: IntuiSec API

Appendix B

IntuiSec TAP Protocol

GET_CONNECTIVITY_INFO: If the device making the request is authorized after exchanging the HomeID, the receiver of the request transfers the serialized connectivity information to the requestor. This command is sent as part of the Easy Setup process.

GET_DEVICE_INFO: Gets information about all UPnP Providers residing on the Fixed Device as a DeviceInfo object in its serialized form. The information consists of the ProviderID, OwnerID, HomeID, ProviderName, and ProviderType. The last parameter corresponds to the UPnP Provider type and is used to display an icon as part of the Trust Builder application.

GET_USER_ID: Gets the UserID of the currently active user on the remote device. Since the current implementation only handles single user devices, this in effect returns the OwnerID of the remote device.

PASSLET <Passlet>: Sends a serialized Passlet to the TAPed device. The Passlets are sent in plain text and consist of: UserID, ProviderID, DeviceName, Expiry-Time, Permissions, and ProviderType. The ProviderType corresponds to the UPnP Provider type (e.g. Printer or Renderer) and is used to display an icon as part of the Passlet Manager application.

CONNECTIVITY_INFO <conn_info>: Sends connectivity information from a

Home Device to a Visitor Device after a Passlet has been sent to that device. As described in Section 4.4.2, a Passlet given to a visitor incorporates additional connectivity information. Thus this command is only used in the case where a device is sending a Passlet to a visitor device. After comparing the home IDs in the beginning, the IRClient infers that the other device is a visitor device. Then after sending the Passlet over with the PASSLET command, the client sends a CONNECTIVITY_INFO command to indicate that it is about to send over connectivity information as well. This contains only one key (as opposed to a list of 100 keys that a home device receives when it gets the connectivity info from an access point).

UNAUTHORIZED: Sent when a visitor device attempts to perform an action that only home devices are authorized to perform. Currently the only time this occurs is when a visitor device attempts to retrieve connectivity information from a home infrastructure device.

ERROR: Signals that an error occurred.

OK: Indicates that a command was processed without error.

CLOSE: Terminates the TAP.

Appendix C

ISCS Protocol

PASSLET <Passlet>: Used when the IntuiSec Client wishes to present a Passlet to be posted to the IntuiSec server. The Passlet is a Passlet in its serialized form. Returns an OK if the post was successful, otherwise returns ERROR (if it was revoked).

REVOKE <ProviderID, PassletID>: Used when the client wishes to present a Passlet to be revoked by the server. The ProviderID is that of the device for which the Passlet was generated - this makes searching for the relevant Passlet faster in the case where there are many Providers running on the same physical device and so being managed by the same IntuiSec Server. Returns an OK if the revocation was successful.

GET_SEC_PARAMS <UserID, HomeID, UDN>: Sent from the client to the server to retrieve the security parameters. The first 3 bytes of the response are:

- Whether or not the user is the owner of the device (if true then 1, else 0).
- Whether or not the device making the request is part of the same home (if true then 1, else 0).
- Whether or not the user has a valid Passlet to access the specified device (if true then 1, else 0).

The next 16 bytes contain the ProviderID of the device with the specified UDN.

This is so that the remote device that made the request can perform the final check, which is whether or not the Provider is on a trusted device.

GET_ACTIONS <UserID, UDN>: Returns a list of semi-colon separated user-level actions that the user is permitted to perform on the specified device. These actions represent high-level, user-perceived functionality and each of them is usually mapped to a number of SOAP-level actions.

ERROR: Signals that an error occurred.

OK: Indicates that a command was processed without error.

Appendix D

Example intuisec.config file

```
// This is the configuration file for a device.  It is
// formatted with the use of structures.  A structure
// consists of a multi-line segment that starts with a tag.
// The INF tag is used for infrastructure devices.  It is
// followed by three lines, the first contains the hardware
// address of the network interface using standard
// hexadecimal separated by colons.  The second contains
// the name of the infrastructure device.  The third
// contains its connectivity type, currently either
// Bluetooth or 802.11.
// The DEV tag is used for Fixed Devices.  It is followed
// by four lines.  The first is the ProviderID; the second
// is the name of the device; the third is the device type
// which is only used for clients to display appropriate
// icons; the fourth is the UPnP Provider UDN
// Here is an example of the two structures that are
// available.
//
// INF:
// 00:0F:3D:38:EC:FC (Hardware address)
// Aristotelis (Device Name)
// Bluetooth (Connectivity Type)
//
// DEV:
// device1 (ProviderID)
// Saad's Printer (Device Name)
// 3 (Device Type)
// 8c073f14-1499-4050-80fe-937dcddf5454 (UDN)
//
// Types of devices:
// Media Server = 1
// AV Renderer = 2
// Printer = 3
```

Appendix E

Abstract Data Types

Connectivity Information

1. Infrastructure Device Name
2. MAC Address
3. Connectivity Type (Bluetooth/802.11)
4. Link Keys

Trusted Device Information

1. ProviderID
2. Provider Name
3. HomeID
4. OwnerID
5. Device Type (for icon display)

Passlet

1. UserID
2. User Name
3. ProviderID
4. Provider Name
5. Expiry Time
6. Permissions
7. Device Type (for icon display)
8. PassletID

Bibliography

[1] Protecting Your Personal Information. Published Online by CBC News, February 2005. www.cbc.ca/news/background/computer-security/wireless.html.

[2] John Tilak. Number of Home Networks to Reach 111m in 2008. *Digital Media News for Europe*, September 2004. www.dmeurope.com/?ArticleID=2941.

[3] D. Kalofonos. IntuiWare Security Functional Specification. Technical report, Nokia Research Center (internal), April 2004.

[4] D. Kalofonos and S. Shakhshir. IntuiSec: a Framework for Intuitive User Interaction with Security in Smart Spaces. Technical report, Nokia Research Center (internal), August 2005.

[5] S. Shakhshir and D. Kalofonos. IntuiSec: a Framework for Intuitive User Interaction with Smart Home Security. Technical Report NRC-TR-2006-003, Nokia Research Center, April 2006.

[6] Digital Living Network Alliance (DLNA) Overview. http://www.dlna.org/about/DLNA_Overview.pdf.

[7] China Tops Cellular Subscriber Top 15 Ranking. Cellular Subscribers Will Top 2B in 2005. Press Release, Computer Industry Almanac Inc., September 2005. www.c-i-a.com/pr0905.htm.

[8] Alma Whitten and J. D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Proceedings of the 8th Usenix Security Symposium*, pages 169–184, 1999.

[9] Mary Ellen Zurko, Charlie Kaufman, Katherine Spanbauer, and Chuck Bassett. Did You Ever Have To Make Up Your Mind? What Notes Users Do When Faced With A Security Decision. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC)*, 2002.

[10] N.S. Good and A. Krekelberg. Usability and privacy: a study of kazaa p2p file-sharing. In *Proceedings of the Conference on Human factors in Computing Systems (CHI)*, pages 137–144, Ft. Lauderdale, Florida, USA, 2003. ACM Press.

[11] P. Dourish, R.E. Grinter, J. Delgato de la Flor, and M. Joseph. Security in the wild: User strategies for managing security as an everyday, practical problem. In *Personal and Ubiquitous Computing*, pages 391–401. Springer-Verlag, 2004.

[12] K-P Yee. User interaction design for secure systems. In *Proceedings of the 4th international conference on information and communications security (ICICS)*, December 2002.

[13] M. E. Zurko and R. T. Simon. User-centered security. In *Proceedings of the New Security Paradigms Workshop*, September 1996.

[14] A. Adams, M. A. Sasse, and P. Lunt. Making passwords secure and usable. In *Proceedings of the HCI'97 conference on people and computers XII*, pages 1–19, August 1997.

[15] A. Adams and M. A. Sasse. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures. In *Communications of the ACM*, pages 40–46, December 1999.

[16] S. Brostoff and M.A. Sasse. Are passfaces more usable than passwords? a field trial investigation. In *Proceedings of the HCI 2000 conference on people and computers XIV*, September 2000.

[17] R. Dhamija and A. Perrig. Deja vu: a user study using images for authentication. In *Proceedings of the 9th USENIX security symposium*, August 2000.

[18] P. Gutmann, D. Naccache, and C. Palmer. Security usability. In *IEEE Security and Privacy*, pages 56–58, July/August 2005.

[19] D. K. Smetters and R. E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *New Security Paradigms Workshop.* ACM, 2002.

[20] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, 2001.

[21] R. De Paula, X. Ding, P. Dourish, K. Nies, B. Pillet, D. Redmiles, J. Ren, J. Rode, and R. Silva Filho. Two experiences designing for effective security. In *Symposium On Usable Privacy and Security (SOUPS 2005)*, Pittsburgh PA, USA, 2005.

[22] P. DiGioia and P. Dourish. Social navigation as a model for usable security. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 101–108. ACM, 2005.

[23] C. Brodie, C.M. Karat, J. Karat, and J. Feng. Usable security and privacy: a case study of developing privacy management tools. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 35–43. ACM, 2005.

[24] Buffalo Technology. *AirStation One-Touch Secure System (AOSS^{TM})*, October 2004. `www.buffalotech.com/documents/pdf/AOSS_WP_Final.pdf`.

[25] Broadcom Corporation. *Securing Home Wi-Fi Networks: A Simple Solution Can Save Your Identity*, May 2005. `www.54g.org/pdf/Wireless-WP200-RDS.pdf`.

[26] S. Meyer and A. Rakotonirainy. A survey of research on context-aware homes. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*, pages 159–168, 2003.

[27] W.K. Edwards and R.E. Grinter. At home with ubiquitous computing: Seven challenges. In *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 256–272. Springer-Verlag, 2001.

[28] J. Heidemann, R. Govindan, and D. Estrin. Configuration challenges for smart spaces, 1998.

[29] Michael H. Coen. Design principles for intelligent environments. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence AAAI*, pages 547–554, 1998.

[30] K. Connelly and A. Khalil. Towards automatic device configuration in smart environments. In *Proceedings of the Fifth Annual Conference on Ubiquitous Computing*, October 2003.

[31] Y. Katsuno and T. Aihara. Autonomic network configuration for networkable digital appliances. In *IEEE Transactions on Consumer Electronics*, volume 51, pages 494–500, May 2005.

[32] J. Wang, Y. Yang, and W. Yurcik. Secure smart environments: Security requirements, challenges and experiences in pervasive computing. In *Experience Workshop on Pervasive Computing*, July 2005.

[33] P.A. Nixon, W. Wagealla, C. English, and S. Terzis. Security, privacy and trust issues in smart environments. In *Smart Environments: Technologies, Protocols and Applications*. Wiley-Interscience, 2005.

[34] R. Campbell, J. Al-Muhtadi, P. Naldurg, G. Sampermane, and M. D. Mickunas. Towards security and privacy for pervasive computing. In *Proceedings of the International Symposium on Software Security*, 2002.

[35] M.J. Covington, W. Long, S. Srinivasan, A.K. Dey, M. Ahamad, and G.D. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the ACM Symposium on Access Control Models and Technologies*, May 2001.

[36] H. Nakakita, K. Yamaguchi, M. Hashimoto, T. Saito, and M. Sakurai. A study on secure wireless networks consisting of home appliances. In *IEEE Transactions on Consumer Electronics*, pages 375–381, May 2003.

[37] J. Seigneur, C. Jensen, S. Farrell, E. Gray, and Y. Chen. Towards security auto-configuration for smart appliances. In *Proceedings of the Smart Objects Conference*, 2003.

[38] D. Balfanz et al. Network-in-a-box: How to set up a secure wireless network in under a minute. In *Proceedings of the 13th Usenix Security Symosium*, pages 207–221, 2004.

[39] D. Balfanz et al. Talking to strangers: Authentication in ad hoc wireless networks. In *Proceedings of the 2002 Network and Distributed Systems Security Symposium*, pages 23–35, 2002.

[40] D. Balfanz; G. Durfee; D. K. Smetters. In search of usable security: Five lessons from the field. In *IEEE Security & Privacy*, pages 19–24, September/October 2004.

[41] U. Holmström. User-centered design of security software. In *Human Factors in Telecommunications*, May 1999.

[42] C. M. Ellison. Home network security. In *Interoperable Home Infrastructure*, November 2002.

[43] UPnP Forum. UPnP Security Ceremonies V1.0. Published Online, October 2003. `www.upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf`.

[44] UPnP Forum. UPnP Device Security V1.0. Published Online, November 2003. `www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf`.

[45] Carl Ellison. Upnp security console v1.0. Published Online, November 2003. www.upnp.org/standardizeddcps/documents/SecurityConsole_1.0cc.pdf.

[46] UPnP^{TM}Forum. *Understanding UPnP^{TM}: A White Paper.* http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc.

[47] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *International Workshop on Security Protocols*, 1999.

[48] Certificate authority. http://en.wikipedia.org/wiki/Certificate_authority.

[49] Nokia Corporation. *Datasheet of the Series 60 SDK for C++.* http://sw.nokia.com/id/1ab7816d-a4cb-4561-b4f8-249ec5a62f15/DS_S60_cpp_SDK.pdf.

[50] Ubuntu linux. http://www.ubuntulinux.org.

[51] Debian linux. http://www.debian.org.

[52] Intel Authoring Tools for UPnP Technologies. http://www.intel.com/cd/ids/developer/asmo-na/eng/downloads/upnp/tools/index.htm.

[53] Symbian. *Symbian OS v7.0s Documentation.* http://www.symbian.com/developer/techlib/v70sdocs/index.asp.

7735-[illegible]