# Security for Rural Public Computing

by

Sumair Ur Rahman

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

## Abstract

Current research on securing public computing infrastructure like Internet kiosks has focused on the use of smartphones to establish trust in a computing platform or to offload the processing of sensitive information, and the use of new cryptosystems such as Hierarchical Identity-based Encryption (HIBE) to protect kiosk user data. Challenges posed by rural kiosks, specifically (a) the absence of specialized hardware features such as Trusted Platform Modules (TPMs) or a modifiable BIOS in older recycled PCs, (b) the potential use of periodically disconnected links between kiosks and the Internet, (c) the absence of a production-ready implementation of HIBE and (d) the limited availability of smartphones in most developing regions make these approaches difficult, if not impossible, to implement in a rural public computing scenario. In this thesis, I present a practical, unobtrusive and easy-to-use security architecture for rural public computing that uses a combination of physical and cryptographic mechanisms to protect user data, public computing infrastructure and handheld devices that access this infrastructure. Key contributions of this work include (a) a detailed threat analysis of such systems with a particular focus on rural Internet kiosks and handheld devices, (b) a security architecture for rural public computing infrastructure that does not require any specialized hardware, (c) an application-independent and backward-compatible security API for securely sending and receiving data between these systems and the Internet that can operate over delay tolerant links, (d) an implementation of my scheme for rural Internet kiosks and (e) a performance evaluation of this implementation to demonstrate its feasibility.

## Acknowledgements

I would like to thank my thesis advisors, Prof. Srinivasan Keshav and Prof. Urs Hengartner, for all their guidance and support throughout my graduate studies at the University of Waterloo; my thesis committee, Prof. Ian Goldberg and Prof. Paul Ward, for all their detailed comments on this thesis; my colleagues at the Tetherless Computing and CrySP Labs for all their advice and moral support; Jessica Miranda and Margaret Towell for all their help; and finally, Prof. Chrysanne DiMarco, whose passion for research inspired me to pursue graduate studies.

## Dedication

*For my loving*
*parents Shafiq and Rubina,*
*brother Saad and sister Eman,*
*whose encouragement,*
*endless support*
*and countless sacrifices*
*made this possible.*

# Contents

# List of Figures

# Chapter 1

# Introduction

Public computing infrastructure, such as rural Internet kiosks, is being widely deployed in developing regions around the world [39, 43] with the goals of providing people in these regions with low-cost access to the Internet [10, 33, 25], improving health care services [17], expanding the reach of e-government services [16], and simplifying the provision of rural microfinance services [12].

To date, most research in this area has focused on the design and implementation of the underlying network infrastructure. Notable exceptions are recent work by Seth *et al.* [32] and Kate *et al.* [13] which propose the use of IBE (Identity-based Encryption) to provide security and anonymity in DTNs (Delay Tolerant Networks). This thesis aims to address the problem of securing rural public computing in depth, taking into account the needs of all stakeholders, the constraint imposed by the hardware and software available in developing regions, and the need to minimize costs through the use of open-source off-the-shelf software tools, to produce a practical, unobtrusive security architecture for rural public computing.

## 1.1 Background

The rural public computing model I consider is based on the KioskNet platform [10] and consists of one or more recycled commodity PCs that connect to the Internet either over a long-range wireless link such as WiMAX, or over a purpose-built DTN (Delay Tolerant Network), as shown in Figure 1.1. (I describe the roles of Kiosk Controllers and Proxy Servers in Section 3.3.) Users in this model may also use recycled mobile devices to wirelessly access the rural public computing infrastructure.

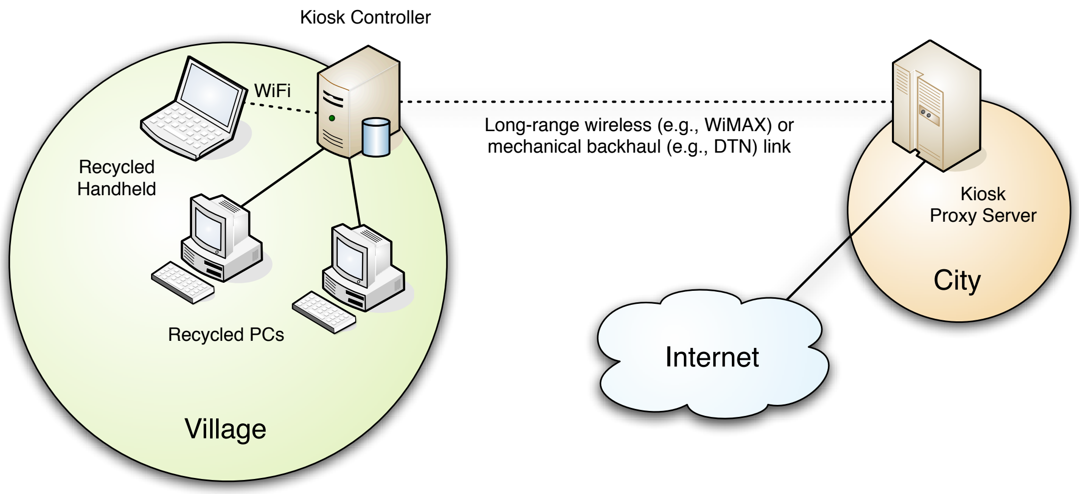

Figure 1.1: Rural Public Computing Infrastructure

Currently, rural kiosks provide weak security and therefore cannot support secure applications such as banking. For example, there is nothing to stop a kiosk administrator from installing malicious software or accessing user data stored on the kiosk. Securing rural Internet kiosks is more challenging than securing privately-owned PCs because users of these kiosks do not own the hardware that they use and cannot always trust kiosk owners/operators. In addition, the potential use of DTN links in remote regions to connect kiosks to the Internet precludes the use of traditional session-oriented technologies such as SSL to build secure applications for kiosks.

## 1.2 Thesis Scope

This thesis focuses on four main areas: (a) providing a detailed security risk assessment of public computing infrastructure that clearly identifies attackers, their capabilities and potential attacks, (b) the design of a suitable security architecture, (c) a security analysis of this architecture and finally, (d) an implementation and evaluation of the same. My goal is to deliver a low-cost implementation that would be practical for use in rural developing regions. Thus, my design makes use of proven, readily available, free open-source software tools, relies on existing cryptosystems and does not require the use of specialized hardware such as Trusted Platform Modules constraints.

In this thesis, I present a practical, unobtrusive and easy-to-use security architecture for rural public computing thats uses a combination of physical and cryptographic mechanisms to protect user data, public computing infrastructure and handheld devices that access this infrastructure. Key contributions of this work include (a) a detailed threat analysis of such systems with a particular focus on rural Internet kiosks and handheld devices, (b) a security architecture for rural public computing infrastructure that does not require any specialized hardware, (c) an application-independent security API for securely sending and receiving data between these systems and the Internet that can operate over periodically disconnected links, (d) an implementation of my scheme for rural Internet kiosks and (e) a performance evaluation of this implementation to demonstrate its feasibility.

## 1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, I review the current state-of-the-art and other related work. In Chapter 3, I present my system model, followed by a comprehensive threat analysis in Chapter 4. I continue in Chapter 5 by presenting my security architecture. Chapter 6 analyses this architecture and its effectiveness in preventing or mitigating the attacks identified in Chapter 4. I then discuss implementation issues in Chapter 7 and evaluate the performance of my implementation in Chapter 8. I conclude and briefly discuss potential directions for future work in Chapter 9.

# Chapter 2

# Related Work

In this chapter, I provide a detailed review of related work, focusing on three specific areas of interest: trusted public computing, security for delay tolerant networks and security mechanisms for mobile devices.

## 2.1 Trusted Public Computing

Prior work in trusted public computing has focused on allowing users to access remote services via an untrusted proxy without revealing sensitive information, such as private keys or passwords, to the proxy. Several schemes [9, 31, 37, 4] propose a user query the proxy about its state and verify that this state is trustworthy via a mobile device trusted by the user. I now briefly review the first three of these approaches as they are the most closely related to my scenario and discuss their feasibility in a rural public computing environment.

Garriss *et al.* propose a scheme that makes use of a mobile device trusted by a user to establish trust in a kiosk providing public computing services [9]. Through the use of software installed on both the mobile device and PC providing kiosk services, the user first uses the camera on his/her mobile device to capture a visual "tag" representation of a hash of the kiosk's public key. (This visual tag is tamper-evident and either affixed to the kiosk itself or the building housing the kiosk.) The mobile device then attempts to authenticate with the kiosk over a short-range BlueTooth wireless link, with the hash value captured via its camera being used to guard against potential man-in-the-middle attacks during this authentication phase. The mobile device then requests that the kiosk reboot in front of the user, upon completion of which it queries the kiosk to verify the integrity of its software stack against a set of known configurations stored on the mobile device. This step makes use of specialized security hardware on the kiosk PC called a Trusted Platform Module (TPM) [42] that is used to achieve trusted boot. Once the user's mobile device has successfully verified the integrity of the kiosk, it gives the kiosk access to any required personal user data (e.g., private keys) stored on the mobile device and informs the user that he/she is free to use the (now) trusted kiosk. Kiosk operators use a similar protocol to verify remotely the integrity of their kiosks over the Internet using a "kiosk supervisor" application. Challenges to applying this scheme in a rural public computing environment include its reliance on specialized hardware not currently available in recycled PCs such as TPMs and newer x86 processors with Dynamic Root of Trust for Measurement (DRTM) capabilities, as well as the requirement that all users carry trusted mobile devices.

Seshadri *et al.* propose a software-based scheme called "Pioneer" [31] that allows for the untampered execution of applications on untrusted legacy platforms. In Pioneer, when a user wishes to run an application on an untrusted platform, an external trusted entity called a dispatcher (e.g., a mobile device trusted by the user) is first used to engage in a challenge-response protocol with the untrusted platform. As part of this protocol, Pioneer establishes a "dynamic root of trust" on the untrusted platform using its "verification function". Any attempt by an attacker to tamper with the verification function increases its execution time beyond expected values, allowing the attack to be detected by Pioneer's dispatcher. After establishing this root of trust on the untrusted platform, Pioneer verifies the integrity of the application the user wishes to run and then executes this software in the untampered execution environment provided by the dynamic root of trust. The main challenges to applying this approach in a rural public computing scenario are a number of open problems noted by Seshadri *et al.* as well as the requirement that dispatchers know all possible hardware configurations of the untrusted platform. (Kiosk terminals, the untrusted platforms in my rural public computing scenario, are recycled PCs supplied by franchisees whose hardware configurations are unknown to the franchisers who operate rural public computing infrastructure.)

Surie *et al.* propose the "Trust-Sniffer" system [37], a scheme which allows users to quickly establish trust in a public computing platform. In Trust-Sniffer, all users carry an inexpensive USB memory stick called a "trust initiator" that contains a minimal trusted OS and hashes of all known trusted OS kernels and applications. When a user wishes to establish trust in an untrusted public computing system using Trust-Sniffer, he/she boots the system with his/her trust initiator. (It is assumed that the PCs serving as public computing platforms in this scheme are capable of booting via USB.) The minimal trusted OS on the trust initiator then verifies the integrity of the public computing platform's on-disk OS kernel against hashes stored on the trust initiator. If the on-disk kernel is successfully validated, the trust initiator reboots the public computing platform using its on-disk OS. The (now) trusted OS on the public computing platform then validates every application the user attempts to run (before these applications are run) by computing hashes of their binaries and comparing these against hashes of known trusted applications stored on the trust initiator. Applications which cannot be verified by the trusted OS are not run and an error message is displayed on the public computing platform to inform the user. When the user has finished using the public computing platform, he/she simply shuts the system down and removes his/her trust initiator. Challenges to applying this approach in a rural public computing environment include the need to update the hashes stored on trust initiators every time the software on public computing platforms is changed and the requirement that public computing platforms be capable of booting via USB. (Recycled PCs serving as kiosk terminals, the platform to be trusted by users in my rural public computing scenario, may not be capable of booting via USB.)

Alternative schemes to those discussed above [6, 24, 35] propose offloading the processing of sensitive information from the untrusted proxy to a mobile device trusted by the user. The main problem with applying these approaches to my scenario is their dependency on all users carrying trusted mobile devices, a requirement which is currently unrealistic in the developing regions where rural public infrastructure would most likely be used.

## 2.2 Security in Delay Tolerant Networks

Delay Tolerant Networks (DTNs) are characterized as networks where end-to-end connectivity is not guaranteed. Typically used in performance-challenged environments, applications of DTNs include space and underwater exploration, defense systems, ad-hoc sensor networks and providing Internet access in remote developing regions.

Several security architectures have been proposed for DTNs. Some focus on the use of light-weight crypto systems such as Identity-Based Encryption (IBE) [3, 13, 32], while others make use of traditional PKI [8, 38]. I now briefly review these approaches.

Farrell *et al.* present an overview of the security analysis of DTNs carried out by the IETF's Delay Tolerant Networking Research Group (DTNRG) [8]. They discuss potential threats against the availability of DTNs such as resource consumption, denial of service and traffic storms as well as threats against the confidentiality and integrity of in-flight data. Farrell *et al.* build on this threat analysis to present a series of security requirements and design considerations, including the use of Public Key Infrastructure (PKI). A suitable key management scheme for DTNs is considered an open problem.

Symington *et al.* extend the work done by Farrel *et al.* to propose the DTN Bundle Security Protocol Specification [38] which details the data confidentiality and integrity services provided by the bundle security protocol. They describe extensions to the DTN Bundle Protocol Specification [30], which like IPSec [14], allows for the authentication of bundles' sources as well as the confidentiality and integrity of payloads carried by these bundles. Symington *et al.* also describe the processing of secure bundles, which as demonstrated in a trial implementation by Scheirer and Chuah [28], results in very poor performance with respect to processing times, even when the host systems are 2.8 GHz workstations. It should also be noted that while Symington *et al.* focus on the secure exchange of messages between DTN nodes and the required format specifications, my concern is at the application and usability level; that is, how users can send and receive secure messages.

Kate *et al.* [13] and Seth and Keshav [32] choose instead to make use of a Hierarchical Identity-Based Encryption (HIBE), a variant of IBE, for their security architectures. HIBE is a light-weight crypto system which, as with IBE, allows users to use their human-readable identities (e.g., email addresses) to generate public keys, eliminating the need to distribute public keys. Whilst technically sound, the main problem with these approaches is the absence of a production-ready implementation of HIBE. I also note that the IBE toolkit formerly distributed at no cost by Voltage Security, Inc. [44] is no longer available for download as of August 2008.

## 2.3 Security for Mobile Devices

Recent interest in mobile computing has resulted in a wealth of literature on securing mobile devices such as laptops, PDAs and smartphones. I now review some of the more relevant work in this area, touching on topics such as the security model for the "One Laptop per Child" (OLPC) project [15], security mechanisms proposed to protect private keys stored on mobile devices [19, 20], the zero-interaction authentication of mobile devices' users [7], a scheme which makes use of smartphones to simplify the access control to shared computing resources [5], a scheme which allows for the secure use of untrusted ubiquitous computing resources [36] and the security architecture for Research In Motion's (RIM) BlackBerry smartphone [2, 27].

Bitfrost [15] is the security architecture for OLPC, a project whose goal is to distribute low-cost, networked laptops (the XO) to children in the developing world, allowing these children to gain computer experience and communicate with other children using XOs. This project's goal is different from that of rural public computing, where people use shared public infrastructure, instead of individual laptops, to (potentially) perform secure transactions, such as online banking. In Bitfrost, a user has full control over his/her laptop, and the security model makes it difficult, but not impossible, for the user (and software) to execute actions which may compromise the security of the laptop. The designers of Bitfrost also have the advantage of being in control of the underlying hardware. For example, they can use the BIOS for boot time integrity checking, which we cannot. Bitfrost's users hold self-signed certificates that bind their name to a public key, the usage of which is transparent to users, as the infrastructure generates and processes certificates on their behalf.

Because secret information stored on mobile devices (e.g. private keys) is typically protected by passwords or encryption keys protected by the same, dictionary attacks against these passwords are a significant threat if the devices are lost or stolen. MacKenzie and Reiter [19, 20] propose a scheme which requires mobile devices to interact with a remote server when performing private key operations such as signing and decryption. The protocols which form the basis of this scheme are designed such that the remote server doesn't need to be trusted and no prior relationship needs to exist between the mobile device and the server. MacKenzie and Reiter prove that through the use of their protocols, the remote server poses no threat to the mobile device in that it cannot learn any information about the mobile device's private key that would allow it to maliciously compute signatures or decrypt messages on the device's behalf. A requirement of this scheme is that the remote server be reachable whenever the mobile device requires use of its private keys, as is the case in interactive protocols such as TLS with client authentication. This assumption is not, however, realistic in a rural setting where connectivity between mobile devices and remote servers may be over a DTN.

To reduce the exposure of sensitive information stored on mobile devices, the cryptographic filesystems employed by these devices are typically configured to frequently prompt users for their passwords (e.g. every time a smartphone's keypad is unlocked.) This improves the security of the device as sensitive information stored on it is only decrypted and available for short periods of time, but makes usage of the system more intrusive, in that users face the inconvenience of having to frequently re-enter their passwords. To work around this problem, Corner and Noble propose a scheme called "Zero-Interaction Authentication" (ZIA) [7]. In ZIA, a user carries a small authentication token that contains the decryption key(s) for the user's mobile device. Users authenticate with the token infrequently (e.g. once a day) but the mobile device authenticates with the token frequently (e.g. every few minutes or every time the user attempts to access sensitive information stored on the mobile device) over a short-range wireless link. This scheme eliminates the need for users to frequently re-enter their passwords and provides the mobile device with the same level of physical security, in that sensitive information stored on the device is only accessible when it is in possession of its owner. Challenges to deploying ZIA in a rural environment include the added cost of authentication tokens, the limited availability of batteries used to power such tokens and the replacement of lost/stolen tokens. It is also not clear what the effects of ZIA are on the battery life of the mobile devices that would implement it.

To unify and simplify access control to virtual and physical resources such as computers and office buildings, Bauer *et al.* [5] propose the "Grey System". Designed as a set of software extensions to smartphones, Grey makes use of an authorization framework based on Proof-Carrying Authorization (PCA) to allow users to wirelessly authenticate with various Grey-enabled systems such as PCs and electronic doors using their Grey-enabled smartphones. Grey gives users the flexibility of being able to temporarily delegate their authority to other Grey users, allowing for example, a friend to temporarily access a user's PC. Secret cryptographic material stored on Grey-enabled mobile devices is protected, in the event of loss or left of the device, using the capture-resilience scheme for mobile devices [20] proposed by MacKenzie and Reiter described earlier. Bauer *et al.* detail an implementation of their system and a pilot deployment at Carnegie Mellon University which covers 150 users and over 60 offices. Their performance evaluation does not, however, appear to take into account the effect on the battery life of mobile devices making use of Grey or the portability of its implementation amongst various mobile platforms. Also, as noted earlier, although my model for rural public computing does include the use of mobile devices, it does not require users to own such devices if they only wish to access the public computing infrastructure through kiosks, making an automated access control scheme which requires mobile devices such as Grey, inapplicable.

Smetters *et al.* [36] propose the Instant Matchmaker, a system which uses mobile devices such as smartphones to securely gain remote access to their personal computing resources (e.g., data stored on a PC) using untrusted local computing resources such as a TV in a hotel room. Because this approach assumes the presence of network connectivity between the local computing resource and remote personal computing resources (to form a VPN connection between the two), it is unfortunately inapplicable in a rural public computing scenario where local and remote personal resources may only be connected via a DTN, preventing the use of interactive protocols such as TLS that are required to create VPN links between local and remote computing resources.

At the time of writing, RIM's BlackBerry [27] is the most popular smartphone in North America, trailing rival Nokia in worldwide sales of similar devices and just ahead of newcomer Apple's iPhone smartphone. The BlackBerry platform consists of handsets (the smartphones themselves) and infrastructure called the BlackBerry Enterprise Solution (BES) [2]. In an enterprise setting, BES is deployed by corporations on their intranets behind a firewall such that the BES gateway maintains an open TCP connection over the Internet to communicate with BlackBerry handhelds (belonging to the same corporation) over a cellular network. For individual telecom subscribers, RIM operates and maintains BES-like infrastructure on behalf of telecoms in data centers at its Waterloo, Ontario campus. Other components of the BES infrastructure, such as the messaging server, interface with existing corporate computing infrastructure on the same network (e.g., a Microsoft Exchange server to provide push email service). Because communication between BlackBerry handsets and the BES gateway takes place over potentially untrusted cellular networks, it is secured using a shared secret key unique to each handset and known only to the handset and its parent BES gateway. This key is refreshed on a regular basis using an interactive protocol. Sensitive data stored on BlackBerry handsets may optionally be secured by enabling the devices' content protection feature which acts as an encrypted filesystem. An ephemeral key derived from the BlackBerry handset user's password is used to encrypt the secret key used by this encrypted filesystem. The primary challenge in deploying a BlackBerry-like system in a rural environment is the potential use of DTN links between mobile devices and public computing infrastructure, which in turn makes the use of interactive protocols such as that currently used to refresh the secret key shared between BlackBerry handhelds and the BES gateway impossible.

# Chapter 3

# System Model

In this Section, I introduce my deployment model for rural public computing, identifying the entities that operate, support or use this computing infrastructure, and then outlining typical usage scenarios involving both rural kiosks and mobile devices.

## 3.1 Overview

Rural public computing infrastructure is deployed over a geographically large rural region to provide people in this region with low-cost, periodically disconnected access to information and services on the Internet. A detailed view of a typical deployment scenario is illustrated in Figure 3.1. Components of this setup include rural Internet kiosks (see Section 3.4), recycled handheld devices (see Section 3.5) and the supporting infrastructure described below in Section 3.3. The individuals and organizations which own, operate and use such a public computing environment are organized into a hierarchy which I describe in the following Section.

## 3.2 Concerned Entities

The following entities have an interest in the correct and reliable operation of the rural public computing infrastructure deployed in their geographic region:

- *Franchisers* – franchisers are public or private organizations that own and operate public computing infrastructure deployed in a particular geographic area.

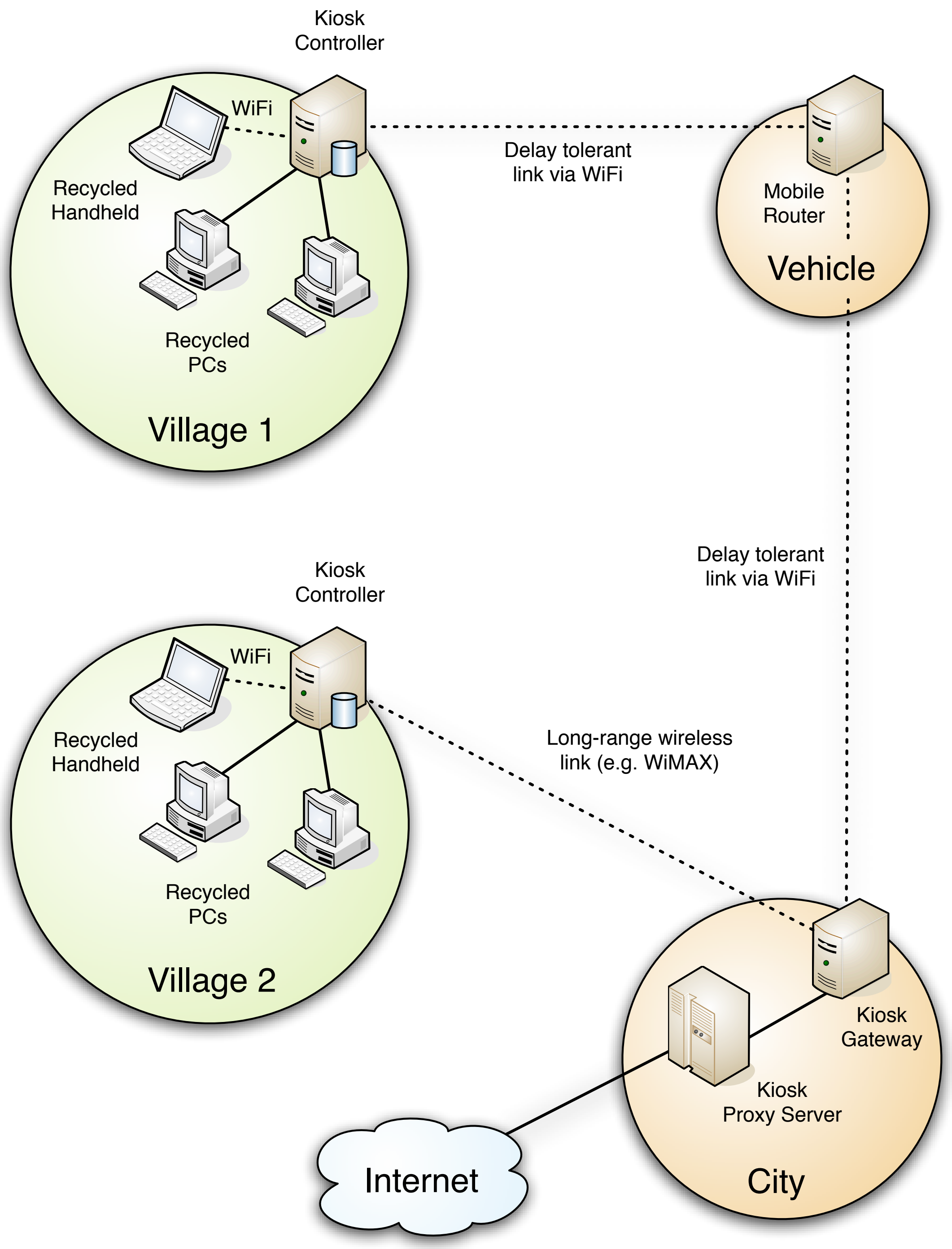

Figure 3.1: Rural Public Computing Infrastructure (detailed view)

- *Franchisees* – franchisees are private organizations or individuals licensed by a franchiser to operate terminals connected to a kiosk controller provided by the local franchiser.

- *Application Service Providers (ASPs)* – application service providers are public or private organizations (e.g., micro finance banks) that are licensed by franchisers to deploy their applications on a rural Internet kiosk.

- *Kiosk Users* – users who subscribe to rural kiosk services with a franchiser, usually through a franchisee, to access services, such as applications provided by ASPs, and the Internet using kiosks owned and operated by local franchisees.

- *Handheld Users* – users who wirelessly access kiosks via handhelds (described in section 3.5). These users would want to be able to send and receive data over the public computing infrastructure deployed in their franchiser's region.

Franchisers control all infrastructure components. This effectively creates a "closed universe" where franchisers have control over all franchisees and registered users in their region, as well as user data and the software running on terminals. I make use of this organizational structure in Chapter 5 when proposing the use of a Public Key Infrastructure (PKI) in my security architecture.

## 3.3 Infrastructure

Each rural public computing deployment provides service to users in a specific geographic region and is independently administered by a *Franchiser* (see Section 3.2). I illustrate a typical deployment scenario in Figure 3.1. Infrastructure components of a typical deployment include:

- *Kiosk Controllers* – servers deployed at rural kiosks that have wired connections to terminals (see below), providing them with network boot, a network file system, user management, and Internet connectivity through kiosk proxy servers (also see Section 3.3 below) via long-range wireless (e.g., WiMAX) or a DTN.

- *Kiosk Gateways* – servers with a WiFi network interface, persistent storage, and live Internet connectivity. Deployed at urban locations with broadband Internet access, gateways collect data opportunistically from mobile routers and stage it in local storage before uploading it to the Internet through a proxy.

- *Mobile Routers* – lightweight embedded devices with a WiFi network interface and persistent storage deployed on vehicles that regularly travel between locations with gateways and rural kiosks. Mobile routers opportunistically communicate with gateways and kiosk controllers to transport data between them.

- *Kiosk Proxy Servers* – servers deployed at data centers that serve as a proxy between gateways and disconnection-unaware legacy servers on the Internet. (Legacy servers in this context provide services such as HTTP, IMAP and SMTP.)

## 3.4 Rural Kiosks

Rural Internet kiosks are owned and operated by entities known as *Franchisees* (see Section 3.2). I illustrate a typical deployment scenario in Figure 3.1. Rural Internet kiosks consist of a single kiosk controller (as described earlier in Section 3.3) and one or more inexpensive recycled PCs called *Kiosk Terminals* that are connected to this kiosk controller over wired Ethernet links.

## 3.5 Handhelds

Handhelds are recycled smartphones or PDAs with one or more wireless interfaces, including cellular technologies such as GSM or CDMA and relatively short-range wireless technologies such as Bluetooth or WiFi and a built-in camera. Although such handhelds may also have access to cellular data services such as EDGE or HSDPA, it is assumed that the cost of using these services in a rural environment is prohibitive. Owners of handhelds would use instead use the devices to wirelessly access rural public computing infrastructure in order to send and/or receive data (in a potentially delay tolerant setting) at a significantly lower cost.

## 3.6 Usage Scenarios

In this section, I briefly present two usage scenarios: the first involving the use of a rural Internet kiosk to conduct the sale of agricultural produce, and the second, involving the use of handhelds to send a video message.

### 3.6.1 Rural Internet Kiosks

In this usage scenario, I describe how Karina, a rural Internet kiosk user, uses a village kiosk to negotiate the sale of her farm's produce using a hypothetical application, *SellProduce*, deployed at the kiosk by a produce wholesaler registered as an ASP.

1. Karina logs into a recycled PC at her local kiosk, launches SellProduce and specifies the produce she has for sale along with her asking price. She then hits the submit button to send her offer to the wholesaler. SellProduce generates an offer message and transmits this to the kiosk controller.
2. The kiosk controller forwards the offer message to a kiosk proxy server which then forwards the message to a kiosk used by the wholesaler. (This may be connected to the proxy directly via the Internet.)
3. The produce wholesaler reviews Karina's offer in SellProduce and responds with another offer message detailing desired purchase price, quantity and timeframe. When the wholesaler submits this offer through SellProduce, the message is sent to the kiosk proxy server.
4. The kiosk proxy server forwards the wholesaler's offer message to Karina's kiosk controller, which stores it for her to review and respond the next time she logs in. Karina would then either respond to further negotiate her terms of sale or confirm the sale as described above.

In the absence of security mechanisms that guarantee the authenticity, integrity and privacy of the offer messages exchanged between Karina and the wholesaler, it would be difficult for either side to trust and act on the messages that they receive. As mentioned earlier, the potential use of a DTN link between kiosks and the kiosk proxy server prevents the use of traditional, session-oriented technologies such as SSL to secure kiosk applications such as SellProduce. Other applications with similar security requirements include rural banking, government record and telemedicine.

### 3.6.2 Handhelds

In the usage scenario below, I describe how Hana, a handheld user, uses a hypothetical application called *VideoMessenger* on her handheld device to record a video message and send it to Karina, a rural Internet kiosk user.

1. Hana launches VideoMessenger on her handheld and records an event in her village using a camera built into the device. Upon completing this, she enters Karina's email address to specify the message's recipient, and hits send. The message is stored on her handheld until the device next connects with the rural public computing infrastructure in her region.

2. Hana visits a location in his village with a rural Internet kiosk and wirelessly connects to the rural public computing infrastructure (a kiosk controller in this case) with her handheld. The device automatically transfers her video message destined for Karina to the kiosk controller.

3. The kiosk controller forwards the video message to a kiosk proxy server which then forwards the message to the kiosk used by Karina.

4. The video message is delivered to Karina's kiosk and displayed on the recycled PC she uses to login to the kiosk.

As in the previous usage scenario, the absence of security mechanisms that guarantee the authenticity, integrity and privacy of the video message exchanged between Hana and Karina would make it difficult for Karina to believe the message actually came from Hana and that it had not been viewed or altered by a malicious third party before she viewed it.

# Chapter 4

# Threat Model

In this chapter, I describe my security design objectives for rural public computing, identify potential attackers, describe their capabilities, and list potential attacks against rural kiosks, handhelds and their supporting rural public computing infrastructure.

## 4.1 Security Goals

In this section, I outline my security design goals for rural public computing, detailing the requirements for rural kiosks, handhelds and the supporting infrastructure.

### 4.1.1 Infrastructure and Rural Kiosks

My overall security goals are to provide the best possible security for users, operators and infrastructure components given the need to minimize costs, the limited processing capabilities of infrastructure components and the recycled PCs used as terminals, as well as the absence of specialized hardware in recycled PCs, such as TPMs or a modifiable BIOS. Specific security goals, in terms of the four entities that use or operate rural Internet kiosks, are as follows:

- *Franchisers* – franchisers want to detect, if not prevent, the misuse of their infrastructure components by any of the concerned entities or outsiders (defined as someone other than one of the concerned entities from Section 3.2).
- *Franchisees* – franchisees want protection against the spread of viruses over and any attacks launched through the public computing infrastructure.

- *ASPs (Application Service Providers)* – depending on the type of service they provide, ASPs may want franchisers to guarantee the integrity of their software when deployed on the public computing infrastructure. Examples of such software might be microfinance systems operated by banks and medical records systems operated by health care agencies.
- *Kiosk Users* – users are primarily concerned with the confidentiality and integrity of their data.

In addition, all of the above entities would be concerned with the availability of the rural public computing infrastructure. In a potentially disconnected environment, this means the period of disconnection from the Internet would be within the timeframe specified by the franchiser.

The physical and cryptographic mechanisms used to achieve these goals are described in Chapter 5 and then evaluated in terms of the protection they provide against potential attacks (see Section 4.2.2) in Chapter 6.

### 4.1.2 Handhelds

Allowing handheld users to access rural public computing infrastructure with their mobile devices requires meeting the following security and usability goals:

1. *Simple Registration* – mobile users should be able to quickly and easily register their handhelds in a specific franchiser's region and obtain the cryptographic credentials necessary to securely send and receive data over the rural public computing infrastructure.
2. *Secure Communication* – all communication between handheld users' mobile devices and the rural public computing infrastructure should be guaranteed integrity and privacy.
3. *Unrestricted Mobility* – handheld users should be able to access all infrastructure operated by their franchiser in order to securely send and receive data over the rural public computing infrastructure.
4. *Unobtrusive, User-friendly Security* – the security mechanisms used to protect handheld users should be unobtrusive and easy to use, having minimal impact on the performance and battery life of users' mobile devices.

I note that these goals are contradictory and that a trade-off is necessary. As with infrastructure and rural kiosks, the security mechanisms used to achieve these goals are described in Chapter 5 and then evaluated in terms of the protection they provide against potential attacks (see Section 4.2.3) in Chapter 6.

## 4.2 Attacker Model

In this section, I describe my attacker model, highlighting potential attack vectors and identifying the resulting threats against rural kiosks, handhelds and the supporting rural public computing infrastructure.

### 4.2.1 Attack Vectors

In addition to outsiders, defined as being none of the entities introduced in Section 3.2, I assume attackers to be either franchisees or users. Franchisers do not appear in our list of attackers because as operators of the system, its correct and reliable functioning is in their best interest. I exclude ASPs under the premise that any software, data and configuration changes produced by these entities will only affect their own software and data, that is, ASPs' software and data is isolated from that of other ASPs. (This would be achieved by running ASPs' software in separate virtual machines on kiosk terminals.) Attackers may possess one or more of the following:

- *Wireless Communication Channel* – the ability to eavesdrop on, inject messages into or jam the wireless communication channel between infrastructure components and handheld devices, given sufficient physical proximity to these systems.
- *Physical Access* – unfettered physical access to infrastructure components in the absence of authorized franchiser personnel, without knowledge of their administrator passwords.
- *Technical Expertise* – the experience and technical expertise required to modify the software or configuration of a Linux-based system, given network-based or physical access.

### 4.2.2 Threats against Rural Kiosks and Infrastructure

Threats against rural kiosks can be categorized as attacks against the confidentiality, integrity or availability of the system. In terms of confidentiality, I am concerned with the privacy of user data and any secret keys stored in their accounts (see Section 5.2.1 for details). For integrity, I am concerned with the integrity of this data as well as the integrity of infrastructure components, recycled PCs used at kiosks and the impersonation of franchiser personnel and kiosk users. For availability, I consider the jamming of wireless links between infrastructure components. (Recycled PCs are connected to kiosk controllers by a wired link.) When combined with potential attackers, these threats give us the attacks numbered 1 through 12 in Figure 4.1. I further classify each attack according to its likelihood. It should be noted the attacks shown in Figure 4.1 are for illustrative purposes only and that this Figure does not represent a comprehensive list of all possible attacks.

The classification of likely and unlikely threat-attacker combinations in Figure 4.1 is based on the capabilities of a particular attacker, the cost of mounting a particular attack, and the potential benefits. For example, a franchiser would be more likely to attempt to modify the configuration of a kiosk controller in order to disable its wireless interface than set up a jamming signal to achieve the same result, given the cost of setting up the jamming signal and the simplicity of disabling a wireless interface.

Threat-attacker combinations that are marked as ignored appear as such because either the cost of mounting the attack exceeds the benefit to the attacker or because a lower-cost attack that achieves the same result is available.

Because I have no control over the underlying delay tolerant network infrastructure for my implementation, I do not consider traffic analysis attacks against application messages exchanged in a rural public computing network (e.g., looking at the source and/or destination of messages), as these are threats I cannot guard against.

### 4.2.3 Threats against Handhelds

Threats against users of handhelds can be categorized as attacks against the confidentiality, integrity or availability of the system. In terms of confidentiality, I am concerned with the privacy of user data and any secret keys stored stored in their handhelds (see Section 5.2.1 for details). For integrity, I am concerned with the integrity of this data as well as the integrity of handheld devices. For availability, I consider the jamming of wireless links between infrastructure components and handhelds. I also classify each attack according to its likelihood. Potential attacks against handhelds appear alongside attackers as threats 13 through 17 in Figure 4.1. As noted earlier, the attacks shown in Figure 4.1 are for illustrative purposes only and that this Figure does not represent a comprehensive list of all possible attacks against handhelds.

| | F | K | H | O |
|---|---|---|---|---|
| **01. User impersonation at kiosk terminals**<br>*Attacker impersonates user identity at terminal to use an application, view private data or escape liability of malicious terminal use* | ■ | ■ | □ | ■ |
| **02. User impersonation at kiosk controller**<br>*Attacker uses Linux PC with root access to connect to kiosk controller, mount NFS-exported /home and view/fabricate users' data* | ■ | ▨ | □ | ■ |
| **03. Viewing/fabrication/modification of user data at kiosk controller**<br>*Attacker logs into kiosk controller as root or removes hard disk and boots with Live CD to view/fabricate users' data* | ■ | □ | □ | ■ |
| **04. Eavesdropping on wireless channel between infrastructure components**<br>*Attacker sniffs packets on wireless channel between infrastructure components to view private data* | □ | □ | □ | ■ |
| **05. Message injection on wireless channel between infrastructure components**<br>*Attacker injects packets on wireless channel between infrastructure components to generate/corrupt data* | □ | □ | □ | ■ |
| **06. Modification of kiosk terminal software stack**<br>*Attacker replaces terminal application with malicious application to enable attack (e.g., login daemon stores passwords)* | ■ | ■ | □ | ■ |
| **07. Modification of kiosk terminal software configuration**<br>*Attacker changes terminal software configuration to enable attack (e.g., disable logging)* | ■ | ■ | □ | ■ |
| **08. Viewing/fabrication/modification of user data transferred between terminal and kiosk controller**<br>*Attacker sniffs/injects packets on wired channel between terminal and kiosk controller to view/fabricate users' data* | ■ | □ | □ | ■ |
| **09. Impersonation of infrastructure components**<br>*Attacker sets up device to impersonate infrastructure components (e.g., mobile routers) to obtain users' data* | □ | □ | □ | ■ |
| **10. Modification of software stack on KioskNet infrastructure components**<br>*Attacker replaces application on infrastructure components with malicious application to enable attack* | ■ | □ | □ | ■ |
| **11. Modification of software configuration on infrastructure components**<br>*Attacker changes infrastructure component software configuration to enable attack (e.g., disable logging)* | ■ | □ | □ | ■ |
| **12. Jamming of wireless channel between infrastructure components**<br>*Attacker jams wireless channel between infrastructure components to prevent the transfer of data* | □ | □ | □ | ▨ |
| **13. User impersonation on handheld**<br>*Attacker impersonates user identity on handheld to view private data or escape liability of malicious handheld use* | □ | □ | ■ | ■ |
| **14. Viewing/fabrication/modification of user data on handheld**<br>*Attacker views, fabricates or modifies data stored on handheld while owner is unaware or device is lost/stolen* | □ | □ | ■ | ■ |
| **15. Eavesdropping on wireless channel betweeen handhelds and infrastructure components**<br>*Attacker sniffs packets on wireless channel between handhelds and infrastructure components to view private data* | □ | □ | □ | ■ |
| **16. Message injection on wireless channel between handhelds and infrastructure components**<br>*Attacker injects packets on wireless channel between handhelds and infrastructure components to generate/corrupt data* | □ | □ | □ | ■ |
| **17. Jamming of wireless channel between handhelds and infrastructure components**<br>*Attacker jams wireless channel between handhelds and infrastructure components to prevent the transfer of data* | □ | □ | □ | ▨ |

■ Possible, likely attack ▨ Possible, unlikely attack □ Ignored, no benefit to attacker or easier attack exists

**F** Kiosk Franchisees **K** Kiosk Users **H** Handheld Users **O** Outsiders

Figure 4.1: Potential threats against Rural Public Computing

As with kiosks and infrastructure, the classification of likely and unlikely threat-attacker combinations in Figure 4.1 is based on the capabilities of a particular attacker, the cost of mounting a particular attack, and the potential benefits. Similarly, certain threat-attacker combinations are marked as ignored for the same reasons described earlier when discussing kiosks and infrastructure. Finally, as noted above for rural kiosks and infrastructure, I do not consider traffic analysis attacks against handhelds for the same reasons.

# Chapter 5

# Security Architecture

In this Chapter, I detail my security architecture for rural public computing, beginning with rural public computing infrastructure, and then moving onto rural kiosks and handhelds. I finish by discussing how this architecture works in the two usage scenarios introduced earlier in Chapter 3.

## 5.1 Infrastructure

Rural public computing infrastructure components are protected against attack using a combination of cryptographic and physical security mechanisms.

For physical security, I assume that kiosk controllers, mobile routers and gateways are equipped with sealed, tamper-evident enclosures. These enclosures would most likely utilize proprietary screws and locks, in addition to sticker seals over removable enclosure panels, similar to those used by vendors of commercial electronics to detect attempts to open the devices. The other physical security mechanism I rely on is the regular inspection of deployed infrastructure components by franchiser field technicians to check for tampering or damage.

In terms of cryptographic security, all infrastructure components are issued unique credentials called *Infrastructure Credentials* by the franchisers that operate them. Infrastructure Credentials consist of a signing key pair generated by the device and a corresponding certificate signed by the franchiser that binds the infrastructure component's identity to its public key. These credentials are installed in each device's `/root` directory by its operating franchiser when it is first deployed, along with the operating franchiser's public key certificate and the certificates of all trusted CAs. (The franchiser's certificate is signed by one of these trusted CAs.) A device uses its credentials to authenticate to other infrastructure components, and to secure communication between infrastructure components.

Administrator privileges on infrastructure components are limited to authorized franchiser personnel. Namely, only the franchiser knows the administrator passwords for its infrastructure components, but not franchisees (e.g., only franchisers can login as "root" on a kiosk controller).

## 5.2 Rural Kiosk

In this section, I introduce my security architecture for rural Internet kiosk, describing how it provides security for kiosk users and the operators of rural public computing infrastructure. I also describe the security mechanisms used to protect terminals deployed at rural kiosks and secure mechanisms available to franchisers to remotely maintain infrastructure components (i.e., authenticated remote shell commands, tamper-evident system logs and digitally signed software updates).

### 5.2.1 User and Operator Security

Entities that use or operate rural kiosks each possess a unique set of *Entity Credentials*. Entity Credentials consist of a signing RSA key pair and an encryption RSA key pair, along with the corresponding X.509 certificates that bind the holder's identity to the public part of each key pair. Kiosk users and franchisees obtain and use their Entity Credentials as described below.

- *Franchisers* – franchisers self-generate their key pairs and then use the public parts of their signing key pairs to obtain certificates signed by a trusted CA such as VeriSign or Thawte, allowing for inter-franchiser authentication. Certificates for encryption keys are then signed using their own signing keys. (In KioskNet, franchisers' signing certificates are signed by the University of Waterloo.) Franchisers' signing key pairs are then used to sign certificates issued to franchiser administrative personnel, licensed franchisees, and ASPs.
- *Franchisees* – franchisees obtain certificates in a similar fashion to franchisers, with the only difference being their signing certificates are signed by their franchiser. Franchisees use their private signing keys to sign certificates for their encryption keys and all signing certificates issued to users registered at their kiosks.

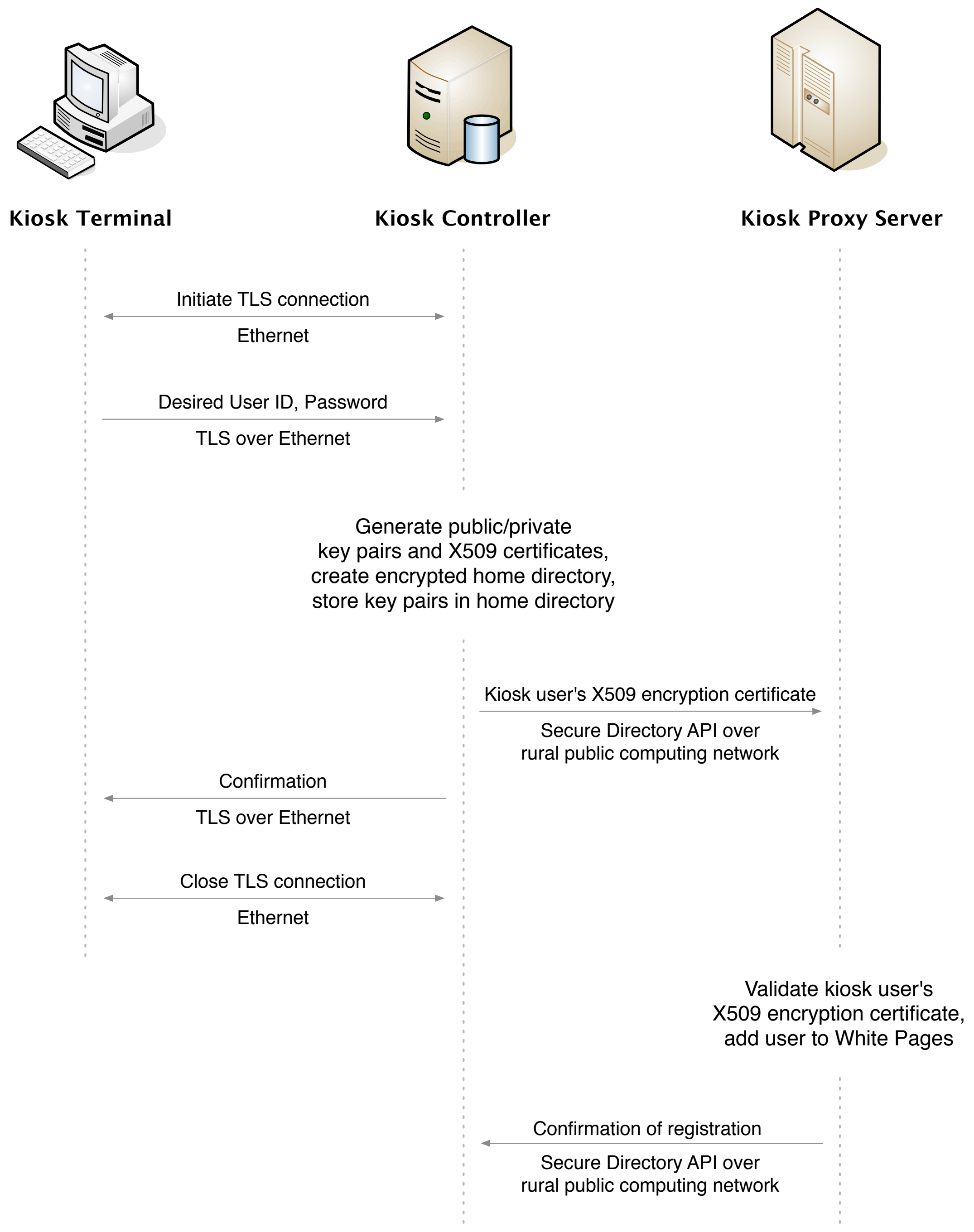
Kiosk Terminal
Kiosk Controller
Kiosk Proxy Server
Initiate TLS connection
Ethernet
Desired User ID, Password
TLS over Ethernet
Generate public/private key pairs and X509 certificates, create encrypted home directory, store key pairs in home directory
Kiosk user's X509 encryption certificate
Secure Directory API over rural public computing network
Confirmation
TLS over Ethernet
Close TLS connection
Ethernet
Validate kiosk user's X509 encryption certificate, add user to White Pages
Confirmation of registration
Secure Directory API over rural public computing network

Figure 5.1: Registration Protocol for new Kiosk Users

- *ASPs* – ASPs obtain certificates from the local franchiser in an identical fashion to franchisees. They use their Entity Credentials to authenticate software deployed at kiosks on their behalf by franchisers and any subsequent updates to this software and to secure the transfer of data between ASPs and users, if necessary.

- *Users* – users obtain their credentials when they register at a rural kiosk using the protocol illustrated in Figure 5.1. (See Section 5.2.2 for a description of the Secure Directory API.) Their signing certificates are signed by the local franchisee, with encryption certificates being signed using their own signing keys. The usage of certificates and key pairs is transparent to users, which is important because previous research has shown that users cannot be expected to manually deal with certificates [11]. Namely, key pairs and certificates are automatically created upon registration and stored in the user's encrypted home directory (see Section 5.2.2). Furthermore, usage of the keys is simplified through the *Secure Directory API* (described in Section 5.2.2), where incoming data is transparently decrypted and verified, and outgoing data is transparently encrypted and signed without any user intervention.

It is noted that as shown in Figure 5.1, a users's key pairs are generated on the kiosk controller, a device operated by the local franchiser. This is done because new accounts are created via a web-based interface accessible through the franchisee's account on any kiosk terminal. (Franchisees are not trusted entities.) Although franchisers are trusted in my attacker model, they may still stand to benefit from replicating users' key pairs outside their accounts and using them to impersonate these users. As such, I leave it to future work to develop a technqiue for establishing trust in a kiosk terminal and moving the creation of new accounts and users' key pairs to a special, trusted guest account on kiosk terminals.

Certificates for users are made available to other kiosk users, franchisers, ASPs, other franchisees and the Internet by means of a public database known as the *White Pages*. This database is maintained by each region's franchiser and updates to it are periodically broadcast to all franchisees (more specifically, the kiosk controllers servicing them) and all licensed ASPs. The database is the only place that is consulted by a kiosk upon receipt of a signed message. Any certificate that no longer shows up in the database is considered revoked, which eliminates the need for a separate certificate revocation mechanism. For a user base of 10,000 with each certificate requiring about 2KB of storage, the entire White Pages database would be around 20MB in size. I note that this size is trivial when considering the hard disk capacity of a kiosk controller or mobile router (at least 40GB), and that mobile routers carried by vehicles can wirelessly transfer upto 100MB of data in a single visit to a kiosk controller.

All certificates described above are chained to a trusted root CA's certificate (e.g., VeriSign, Thawte or the University of Waterloo) such that trusting this certificate alone is sufficient to verify the above entities' certificates. This way, an ASP can, but does not have to, delegate identity verification to a franchisee or even a franchiser, which can be important in rural public computing environments.

### 5.2.2 Terminal Security

Terminals are PCs, typically recycled, that network boot from a read-only image stored in a kiosk controller. These images contain the Linux kernel used by terminals, configuration files and applications. Because we expect a terminal's disk to fail or be corrupted by viruses, user data is stored in the kiosk controller. All applications launched by the user are run on the terminal for better performance. To prevent an attacker from impersonating a user, every user is assigned a Linux login password during registration and has to enter this password into the terminal when logging in. This password is also used for protecting a user's data, which I describe in the following subsection. Terminals automatically shut down when users logout, effectively forcing all kiosk terminals to be rebooted between users, killing any processes the previous user may have left running. Rebooting a Kiosk terminal also allows the BIOS to clear the RAM, making it harder for attackers to launch a memory dump attack on the system to obtain the secret keys used to encrypt users' home directories (refer to Chapter 7 for details on how users' home directories are protected).

Additional physical security mechanisms, such as informative posters and unannounced "surprise" inspections by franchiser personnel, are used to protect terminals against attacks such as hardware key loggers, "shoulder surfing" and terminals booted off malicious terminal images. Anti-virus software is used to protect kiosk terminals against viruses, with virus definitions for this software being regularly updated using digitally signed software updates (see Section 5.2.3).

#### Secure Directory API

The *Secure Directory API* allows developers to easily produce applications that communicate securely between kiosks, handhelds and the Internet. To simplify the development of applications, the Secure Directory API is implemented as a daemon that watches a configured set of directories for new files, providing applications with an API similar to that used by Plan 9 [26]. On kiosk terminals, for example, applications write outgoing data to a user's `~/application/supload` directory and read incoming data from the `~/application/sdownload` directory, where `~/` corresponds to the mount point on the terminal for the user's encrypted home directory.

For incoming data, a daemon implementing the Secure Directory API automatically decrypts and verifies signatures on received data using the user's private keys (stored in his/her encrypted home directory if a kiosk user) and other users' public keys included in senders' signing certificates attached to messages and the White Pages database (introduced in Section 5.2.1) and places it in the `sdownload` directory. Similarly, this daemon automatically encrypts and signs all outgoing data placed in `supload` after looking up the recipient's public encryption key in the White Pages database (the daemon learns the identity of a message's recipient from the accompanying metadata file, as described in Section 7.4.4). Messages destined for a server reachable over the Internet via the kiosk proxy server are either encrypted with the proxy's public encryption key and then forwarded in plaintext over the Internet by the proxy, or encrypted with a specific ASP's public encryption key (in the case of an ASP's server on the Internet). Similarly, messages destined for rural users whose public keys could not be found in the White Pages database (for example, because a White Pages update has not yet propagated to a kiosk controller), are re-addressed and encrypted for the proxy, which then decrypts and encrypts the messages for their intended recipients using its up-to-date copy of the White Pages database. (Because franchisers are assumed to be trusted in my threat model, as described in Chapter 4, it is assumed that all proxy servers run by a user's franchiser are also trusted.)

**Encrypted User Home Directories**

All user data, such as a user's pictures and emails, are stored in kiosk controllers and exported over NFS for access via terminals connected to a kiosk controller. As highlighted in Section 4.2.2, this setup makes it possible for an attacker to connect to the kiosk controller using a Linux PC with administrator access to override filesystem permissions and access the NFS-exported user data or, in a more extreme scenario, to break into the kiosk controller, remove its hard disk, and boot it in a PC with a Live CD to achieve the same.

To protect user data stored in kiosk controllers, users' home directories are created in encrypted virtual volumes. Users' virtual volumes are exported in their encrypted form to terminals over NFS for automatic mounting and decryption when users logs in. The process is reversed when users log out. My implementation is based on off-the-shelf, open-source software, as described in Chapter 7.

In the event a user forgets his/her password, an encrypted backup copy of the key used to encrypt the user's virtual volume is maintained by the franchiser. Users who forget their passwords must contact their local franchiser to request a password reset and the release of the backup copy of their virtual volume's encryption key.

### 5.2.3 Digitally signed Software Updates

Digitally signed software updates are produced and distributed by authorized franchiser administrative personnel. To authenticate these updates, administrative personnel sign them with a designated private signing key, part of an Entity Credential as described earlier in Section 5.2.1, attaching the corresponding public key certificate signed by the franchiser. Signatures on software updates are automatically verified by infrastructure components before the update is applied. The signing certificate chain attached to each update is verified with the franchiser's public key, which is installed when an infrastructure component is first set up. This same mechanism can be used to update the franchiser's public key, if necessary.

It is noted that the loss of a franchiser's private signing key will effectively invalidate all certificates issued to franchisees, ASPs, infrastructure components and users in the franchiser's region. To guard against this, franchisers are expected to maintain one or more secure backup copies.

### 5.2.4 Authenticated Remote Shell Commands

Lukac *et al.* propose the Disruption Tolerant Shell (DTS) [18], a reliable asynchronous remote shell interface which allows for remote system management and configuration. In a rural public computing environment where infrastructure components such as kiosk controllers may be deployed in remote areas, such (authenticated) remote shell commands allow franchiser administrative personnel to maintain these systems remotely (e.g., by running scripts to diagnose a fault) even when the links between these systems and the rural public computing network are delay tolerant. As with software updates, infrastructure components verify the signatures on remote shell commands before running them. Authenticated remote shell commands are produced and distributed in an identical fashion to signed software updates, as described in Section 5.2.3 above.

### 5.2.5 Tamper-evident System Logs

System logs produced by infrastructure components contain valuable debugging information, error reports, notifications of software updates, records of shell commands executed by the administrator, and fingerprints of all executables present on the system. These logs are periodically sent to franchiser administrative personnel, allowing them to monitor deployed infrastructure components. To avoid being noticed, an attacker making changes to the software stack or configuration of an infrastructure component would likely attempt to modify the device's system logs. Thus, these logs need to be made tamper-evident. Tamper-evident protection is provided by means of the hash-chaining scheme detailed in Section 7.4.3.

## 5.3 Handhelds

In this section, I introduce my security architecture for handhelds in a rural public computing scenario. I describe mechanisms for the remote registration of new handheld users, a protocol to authenticate handhelds to rural public computing infrastructure that makes use of visual tags and mechanisms for secure communication between handhelds and infrastructure components.

### 5.3.1 User Registration

The registration of new handheld users is a three-step process:

1. *Register with Franchiser* – handheld users begin by registering with a franchiser through a website. This involves providing the new user's name, contact information (e.g., a phone number or mailing address), specifying the desired user ID (alternatives could be suggested by the franchiser if the desired ID is not available, as is common with most free web-based email services) and providing some identifying information (e.g., a credit card number) that proves the user is real and does in fact correspond to the provided contact information. Upon successful completion of this step, the franchiser provides the new user with confirmation of his/her new user ID and an *Activation Password*. (Activation passwords are randomly-generated alphanumeric strings, unique to each user that are stored in a database maintained by the franchiser alongside the corresponding user's ID.)

2. *Download Handheld Software Package* – in this second step, handheld users download the *Handheld Software Package*, a suite of applications that allow users to

generate public/private key pairs, register with their franchiser and connect to their local rural public computing network. For additional security, the package could be signed with a franchiser's private signing key and verified using the corresponding public key certificate which in turn would be signed by a trusted CA such as VeriSign or Thawte. (In my implementation, the University of Waterloo signs franchisers' certificates.)

3. *Run Registration Client* – after installing the Handheld Software Package, handheld users launch the included *Registration Client* on their mobile device. This application asks the user to enter his/her user ID and activation password (obtained as described earlier) as well as a login password which the user would be required to enter every time he/she connects to the rural public computing network. The registration application then connects to a *Handheld Registration Server* operated by the franchiser over the Internet and completes registration, storing the user's private keys and corresponding public key certificates on the handheld.

I note that it is assumed that no more than one user would be registered per handheld and that users will be able to access the Internet with their handhelds to complete the registration process described above.

When storing private keys on the user's handheld, the Registration Client encrypts them with an ephemeral key derived from the user's password and a salt value. This password has to be entered by the user every time the user connects to the rural public computing network (or every time the private key is used, for added security), ensuring a lost or stolen device cannot be used to easily obtain its owner's private keys.

The protocol implemented by the Handheld Registration Client and Server appears in Figure 5.2. When verifying a Certificate Signing Request (CSR), the Registration Server checks to see that the provided Activation Password matches the supplied user ID and that the signature on the CSR is valid (i.e., that it matches the requesting handheld user's public signing key).

### 5.3.2 Secure Connectivity

To guard against man-in-the-middle attacks in the authentication phase between infrastructure components and handhelds, we require an out-of-band mechanism to bind infrastructure components' public keys to the physical devices. As in the scheme proposed by Garriss *et al.* [9], I make use of printed 2D barcodes called Quick Response (QR) tags [1] that can encode upto 4,296 alphanumeric characters to achieve this. It is assumed that handheld users wishing to connect to an infrastructure component will

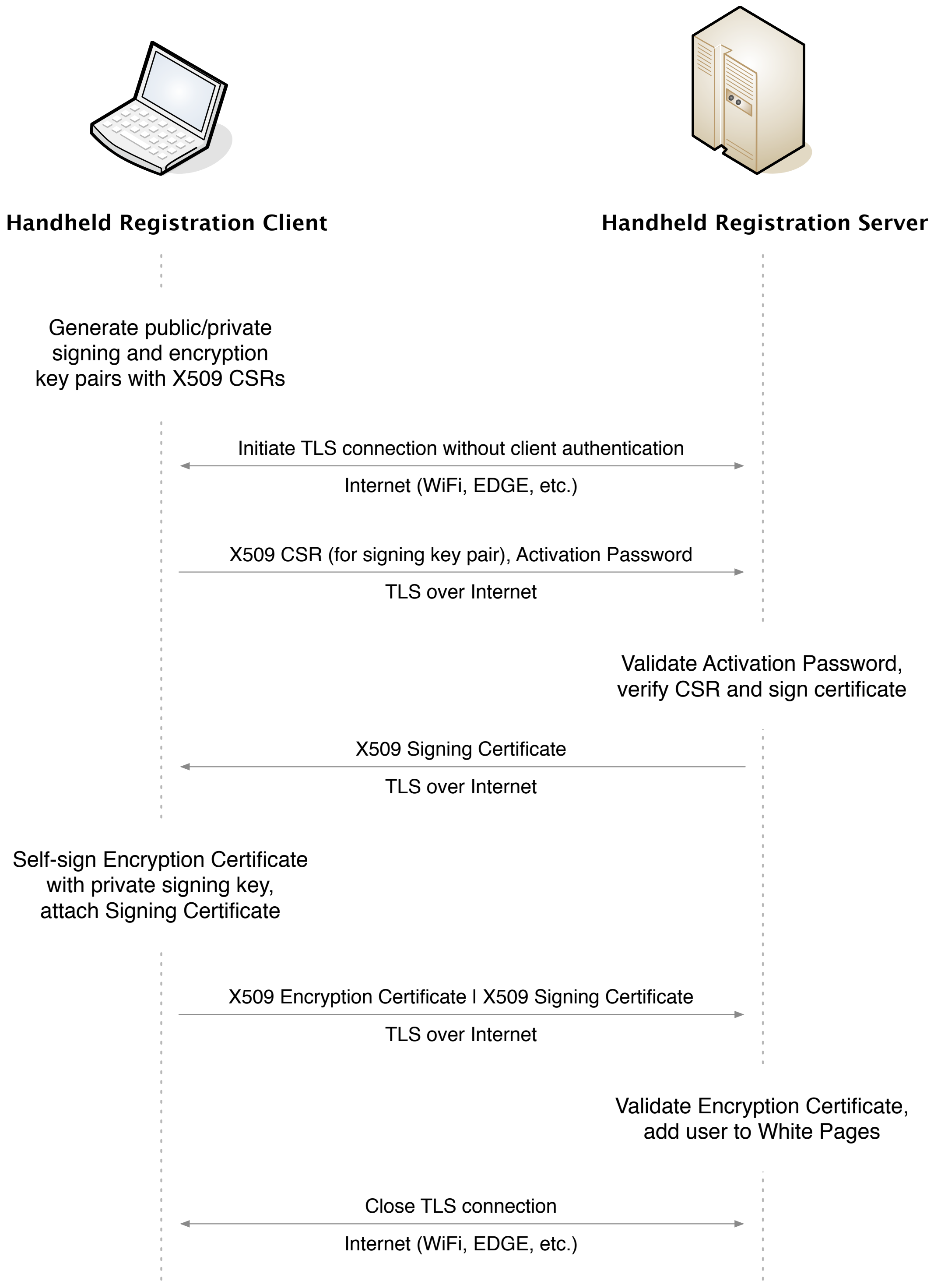
Handheld Registration Client
Handheld Registration Server
Generate public/private signing and encryption key pairs with X509 CSRs
Initiate TLS connection without client authentication
Internet (WiFi, EDGE, etc.)
X509 CSR (for signing key pair), Activation Password
TLS over Internet
Validate Activation Password, verify CSR and sign certificate
X509 Signing Certificate
TLS over Internet
Self-sign Encryption Certificate with private signing key, attach Signing Certificate
X509 Encryption Certificate | X509 Signing Certificate
TLS over Internet
Validate Encryption Certificate, add user to White Pages
Close TLS connection
Internet (WiFi, EDGE, etc.)

Figure 5.2: Registration Protocol for Handheld Users

Figure 5.3: QR Tag encoding SHA1 Hash of X509 Certificate

be within visual range of the device and that the QR tag will be in an area which is sufficiently lit in order to allow a camera in the user's handheld to capture the tag.

Each infrastructure component has a QR tag printed on a tamper-evident sticker attached to it, encoding a hash of its public key, such as that shown in Figure 5.3. (The sample QR tag in Figure 5.3 encodes a SHA1 hash of the X509 certificate in Appendix A.1.) This is similar to the visual tag schemes proposed by Scott *et al.* [29] and McCune *et al.* [21]. These QR tags are generated by the corresponding franchiser and printed on tamper-evident stickers such as those used to seal the enclosures of electronics for warranty purposes, making it difficult for attackers to replicate them.

To connect to the rural public computing network (via an infrastructure component), handheld users launch the *HandheldConnectClient* application on their handheld devices. This application attempts to connect to a daemon running on the infrastructure component, the *HandheldConnectServer*, using a predefined IP address and port number. Following the protocol shown in Figure 5.4, HandheldConnectClient first asks the user to point the handheld's camera at the infrastructure component's QR tag, allowing it to capture a hash of the infrastructure component's public key encoded in the QR tag. HandheldConnectClient then negotiates a connection with the infrastructure component secured by TLS, after verifying that the public key provided by HandheldConnectServer in the TLS server authentication phase matches the hash captured earlier from the infrastructure component's QR tag. The client authentication phase in TLS is extended so the infrastructure component only admits handhelds that are registered with its franchiser.

Handheld users disconnect from infrastructure components through the HandheldConnectClient application. A timeout based on a period of inactivity is used to automatically close the connection if users forget and just walk away.

I note that it isn't necessary to use an out-of-band channel (such as QR tags) to bind infrastructure components' public keys to the physical devices. An alternate scheme would be to simply display an infrastructure component's certificate on a handheld device and ask the user to approve before continuing with authentication (this might involve comparing the infrastructure component's ID in its certificate with the kiosk's name). Recent research, however, shows that users can't be expected to manually deal with certificates [11]. In a rural environment where most users may not understand such certificates, the use of QR tags as described above also serves to simplify usage of the system.

### 5.3.3 Secure Communication

Secure communication between handheld users and other users of the rural public computing infrastructure, including kiosk users and other handheld users, is protected using the Secure Directory API described earlier in Section 5.2.2.

A Secure Directory API daemon running on the handheld device obtains public keys for recipients from its cache of the White Pages database. When the recipient's public encryption key is present in the handheld's cache, the process of sending a message works in the same way described in Section 5.2.2. Because handhelds may have limited opportunities to synchronize their caches of the White Pages database with infrastructure components, it is possible that the handheld may not have the public encryption key for a recipient. In such cases, the Secure Directory API daemon encrypts the outgoing message for the regional kiosk proxy server and forwards the message to this server when the handheld next connects to the rural public computing network. Upon receiving this message, the kiosk proxy server decrypts it using its private encryption key and then encrypts it again using the recipient's public encryption key (the kiosk proxy server maintains the master copy of the White Pages database) before forwarding the message to the recipient. The next time the handheld connects to the rural public computing network, it asks the host infrastructure component for public encryption keys of recipients it didn't have these keys for, storing them in its cache of the White Pages database.

Messages destined for handheld users are flooded throughout the rural public computing network, allowing handheld users to obtain incoming messages by connecting to any infrastructure component in their region (assuming sufficient time for these messages to propagate to the infrastructure component the handheld connects to).

I note that one potential issue with the scheme described above is that encryption of data while the user is interacting with the handheld will likely have a noticeable impact

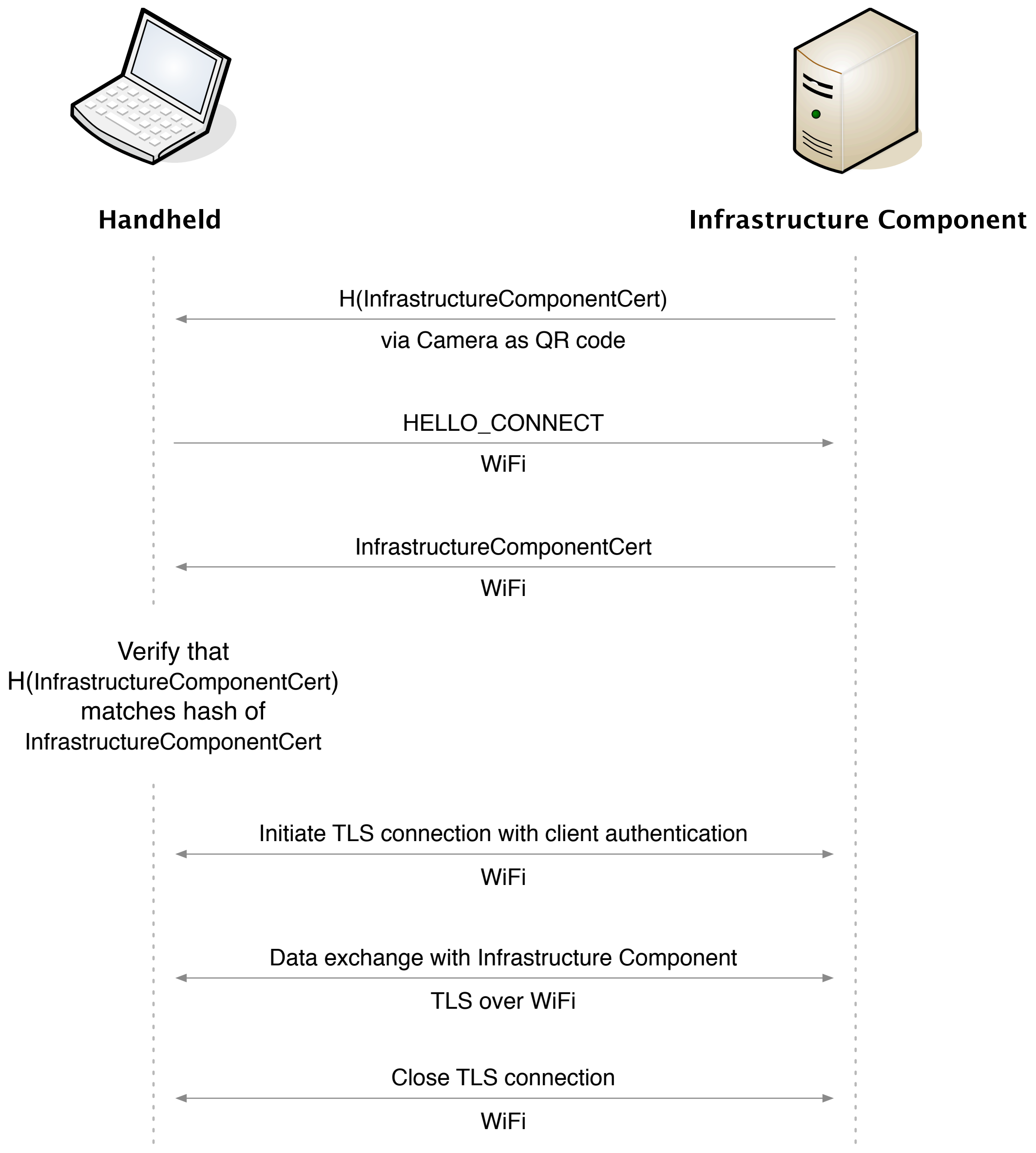
Handheld
Infrastructure Component
H(InfrastructureComponentCert)
via Camera as QR code
HELLO_CONNECT
WiFi
InfrastructureComponentCert
WiFi
Verify that H(InfrastructureComponentCert) matches hash of InfrastructureComponentCert
Initiate TLS connection with client authentication
WiFi
Data exchange with Infrastructure Component
TLS over WiFi
Close TLS connection
WiFi

Figure 5.4: Protocol to establish connection between Handhelds and Infrastructure

on the device's performance, particularly in terms of UI responsiveness. Ideally, all such processing should be completed when the device is not in use (i.e., when it is in a "locked" state). Because the implementation of my security architecture on handhelds is incomplete, I leave addressing this issue as a topic of future work.

### 5.3.4 Secure Storage

To protect user data stored on handhelds in the event the devices are lost or stolen, all user data is encrypted when stored in the devices' flash memory. This is achieved by means of an encrypted file system keyed with a salted ephemeral key derived from the user's handheld password.

## 5.4 Security Architecture Usage Scenarios

In this section, I discuss how my proposed security architecture works for rural Internet kiosks and handhelds, in keeping with the two usage scenarios outlined in Section 3.6.

### 5.4.1 Rural Internet Kiosks

For this usage scenario, I refer back to the rural Internet kiosk usage scenario described in Section 3.6. It is assumed that the produce wholesaler Karina wishes to do business with is registered as an ASP with her regional franchiser.

When Karina hits submit in SellProduce, the application writes her offer message to her home directory in `~/sellproduce/supload`. The Secure Directory API daemon then signs the message with Karina's private signing key stored in her encrypted home directory, attaches her public signing key certificate and encrypts the message for the wholesaler, by looking up the wholesaler's public encryption key in the White Pages database. The Secure Directory API daemon sends the secure offer message to the kiosk controller for forwarding to the kiosk proxy server. (I note that the link between the kiosk controller and proxy server may either be DTN or long-range wireless, as illustrated in the sample network topology in Figure 3.1.) The proxy server then forwards the message to the wholesaler's kiosk controller.

A Secure Directory API daemon running on the wholesaler's computer (this could be a special "terminal" only used by the wholesaler) decrypts the offer message, validates Karina's public signing key certificate, verifies the signature on her message and then passes the validated offer message up to SellProduce running on the wholesaler's computer. Offer messages sent by the wholesaler to Karina would be secured in the same fashion and stored in her encrypted home directory.

### 5.4.2 Handhelds

For this usage scenario, I refer back to the handheld usage scenario described in Section 3.6. It is assumed that both Hana and Karina are already registered users.

When Hana has finished recording the event in her village with VideoMessenger on her handheld, she enters Karina's email address as a recipient and hits send. VideoMessenger creates a message addressed to Karina from Hana and places it in the `/var/apps/videomessenger/supload/` directory on his handheld. A daemon implementing the Secure Directory API running on Hana's handheld detects this file, reads the file's metadata to determine Karina is the recipient and pulls Karina's public encryption key from its local cache of the White Pages database. The outgoing video message is first signed using Hana's private signing key and then encrypted using Karina's public encryption key.

The next time Hana connects to the rural public computing network, her handheld delivers the video message destined for Karina to the infrastructure component he connects to (e.g., a kiosk controller in his village). This infrastructure component then forwards the message to the regional kiosk proxy server (this connection may be over a DTN, please refer to Figure 3.1 for possible network topologies). The kiosk proxy server then forwards the message to Karina's kiosk controller, where the message is stored for the next time she logs in.

When Karina logs into a terminal at her local rural Internet kiosk, a Secure Directory API daemon first decrypts the video message from Hana and then verifies that the signature on the message matches Hana's public signing key (attached to the message itself as part of an X509 certificate). If the signature is successfully verified, the video message is placed in the appropriate sub-directory in Karina's encrypted home directory for her to view the next time she launches the VideoMessenger application on her terminal.

If, alternatively, Karina were a handheld user, her handheld would automatically download Hana's video message when she connects to an infrastructure component (e.g., the kiosk controller in her village) and then follow the reverse process to that described above for sending a message from a handheld to verify the signature on the message and decrypt it for Karina to view on her device.

# Chapter 6

# Security Evaluation

As outlined in Chapter 4, specific security goals, in terms of the entities that use or operate rural kiosks and handhelds, are as follows: (a) franchisers want to detect, if not prevent, the misuse of their infrastructure components, (b) franchisees want protection against the spread of viruses over and any attacks launched against their kiosks, (c) depending on the type of service that they provide, ASPs may want franchisers to guarantee the integrity of their software when deployed on rural kiosks, where examples of such software include tax payment and land registry systems operated by the government, and (d) both kiosk and handheld users are concerned with the confidentiality and integrity of their data. It is noted that franchisera is considered a trusted entity and therefore not the source of any potential attacks. I now describe how the mechanisms proposed in Chapter 5 achieve these goals.

## 6.1 Rural Kiosks and Infrastructure

In this Section I present an analysis of the security mechanisms used to protect rural Internet kiosks and infrastructure components. Figure 6.1 summarizes the security mechanisms I propose and shows how these are combined to guard against the attacks that were presented earlier in Chapter 4. The attack numbers along the top of the table correspond to attacks presented earlier in Figure 4.1. I now describe how these mechanisms defend kiosks and infrastructure components against attacks by users, franchisees and outsiders.

|  | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Password-protected user accounts | ▨ |  |  |  |  |  |  |  |  |  |  |  |
| Encrypted user home directories |  | ▨ | ▨ |  |  |  |  | ▨ |  |  |  |  |
| Root priviledges limited to franchisers |  |  | ▨ |  |  |  |  |  |  | ▨ | ▨ |  |
| In-flight user data signed & encrypted |  |  |  | ▨ | ▨ |  |  | ▨ | ▨ |  |  |  |
| Read-only terminal boot images |  |  |  |  |  | ▨ | ▨ |  |  |  |  |  |
| Unique infrastructure credentials |  |  |  |  | ▨ |  |  |  | ▨ |  |  |  |
| Unique entity credentials | ▨ |  |  |  |  |  |  |  |  |  |  |  |
| Sealed infrastructure component enclosures |  |  | ▨ | ▨ |  |  |  |  | ▨ | ▨ | ▨ |  |

▨ Security mechanism mitigates/prevents attack

□ Security mechanism does not mitigate/prevent attack

Figure 6.1: Security Mechanisms for Rural Public Computing Infrastructure

### 6.1.1 Users

To prevent users from impersonating other users, each user is assigned a password. The operating system running on the terminal ensures that a user enters this password before granting him/her access. For additional security, a terminal is shut down when a user logs out, killing any processes that the user might have left behind and that could use up CPU or memory resources. The next user must then boot the terminal himself/herself, making it harder for franchisees to launch the phishing attack for users' login passwords described in Section 6.1.2 below.

### 6.1.2 Franchisees

The most likely attacks (in terms of simplicity) against terminals and kiosk controllers by franchisees involve tampering with the devices' software stacks and credentials or the impersonation of kiosk terminals to launch phishing attacks for users' login passwords. In a more sophisticated attack, franchisees may use hardware key stroke loggers connected to kiosk terminals in order to obtain users' login passwords.

Physical security mechanisms and the fact that only franchisers can login as "root" on infrastructure components, (as described in Section 5.1,) prevent franchisees from tampering with the software and data stored in kiosk controllers (e.g., adding malicious terminal software, replacing the certificate identifying a franchiser, or extracting the kiosk controller's private key.)

Preventing franchisees from setting up fake login screens on kiosk terminals to obtain users' login passwords is a more challenging problem. The only robust solution is shutting down a kiosk terminal when a user logs out and training users to boot the kiosk terminal before logging in. (Instructional posters displayed at rural kiosks can be used to remind users to log out when the leave, to boot terminals themselves and to check for any irregularities in the connections between kiosk terminals and keyboards that might indicate the presence of a hardware key stroke logger.) Currently proposed techniques for verifying the integrity of public computing platforms such as kiosk terminals [9] require the use of trusted mobile computing devices and the presence of specialized hardware in kiosk terminals (e.g., TPMs), assumptions which will likely not be reasonable in developing regions for quite some time.

Finally, franchisees may also launch phishing attacks against kiosk users by booting terminals with modified (malicious) terminal images. Such attacks can be guarded against by franchisers signing terminal images and forcing terminal boot loaders to verify these signatures before booting. An alternative solution might be to make use of a technique similar to that proposed by Surie *et al.* [37] and have a user boot a terminal from a trusted USB memory stick before logging in. A minimal trusted OS on the stick would then download the kiosk terminal image from the controller, verify its integrity, and, if successful, boot into the kiosk terminal environment. Because most recycled PCs do not support booting via USB and the boot loader features required to verify signatures on terminal images are not currently available in off-the-shelf software, I leave protection against this form of attack to future work.

### 6.1.3 Outsiders

We also need to defend against attacks on a kiosk by outsiders. A terminal is connected to its controller by a wired connection, which makes interception or man-in-the-middle attacks by an outsider difficult. If these attacks are a concern, the boot process could be extended such that a terminal authenticates the controller and downloads the kernel image over a secure connection. However, current off-the-shelf network boot software does not support this feature.

## 6.2 Handhelds

In this Section I present a brief analysis of the security mechanisms used to protect handhelds. Figure 6.2 summarizes the security mechanisms I propose and shows how these are combined to guard against the attacks that were presented earlier in Chapter 4. The attack numbers along the top of the table correspond to attacks presented earlier in Figure 4.1. I now describe how these mechanisms defend handhelds against attacks by outsiders and other handheld users.

| | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|
| Password-protected handhelds | ■ | □ | □ | □ | □ |
| Encrypted storage on handhelds | □ | ■ | □ | □ | □ |
| In-flight user data signed & encrypted | □ | □ | ■ | ■ | □ |
| Unique handheld user credentials | ■ | □ | □ | □ | □ |

■ Security mechanism mitigates/prevents attack □ Security mechanism does not mitigate/prevent attack

Figure 6.2: Security Mechanisms for Handhelds

### 6.2.1 Users

Attacks against handheld users by other handheld users involve the impersonation of handheld users and viewing, modification or fabrication of data stored on handheld devices. Password protection of handheld devices makes it difficult (but not impossible) for other users to access any secret information stored on a device that doesn't belong to them. The encryption of data stored on handheld devices, makes it difficult for attacks to view, modify or fabricate this data in the event the device is lost or stolen. Use of the Secure Directory API to protect data exchanged between handheld devices and the rural public computing infrastructure, as described in Section 5.2.2, ensures the authenticity, integrity and privacy of all such in-flight data.

### 6.2.2 Outsiders

Attacks against handheld users by outsiders involving the impersonation of such a user, the viewing, modification or fabrication of and secret information stored on handheld devices and attacks against in-flight data exchanged between handhelds and the rural public computing infrastructure are mitigated using the same mechanisms described earlier in Section 6.2.1. Attacks against the availability of the system, namely the jamming of wireless channels used for communication between handhelds and infrastructure, are considered an open problem. The use of QR tags to encode the public keys of infrastructure components mitigates man-in-the-middle attacks against handhelds and infrastructure components such as those identified by Garriss *et al.* [9].

Chapter 7

# Implementation

In this chapter, I provide details of my implementation. I begin by detailing the constraints facing my implementation and then describe my implementation for infrastructure components and rural kiosks. Finally, I briefly discuss the issues I encountered. I note that an implementation of my security architecture for handhelds is incomplete and thus, is not described in this Chapter.

## 7.1 Constraints

Constraints for the implementation of my security architecture for rural public computing, as briefly noted in Section 1.2, include the following:

- *Minimize Costs* – the need to minimize the cost of required hardware and software, making use of recycled hardware that will be readily available in developing regions and building my software on free, open-source platforms and libraries to eliminate software licensing fees.
- *Minimize Required Computational Resources* – the use of recycled PCs for kiosk terminals and inexpensive, low-power embedded devices for infrastructure components requires an effort to minimize the required computational resources (i.e., processor and memory) and avoiding the use of specialized hardware features such as TPMs and modifiable BIOSs.

## 7.2 Building Blocks

In this Section, I outline the software building blocks for my implementation, specifically the underlying network infrastructure and the cryptography libraries I used.

### 7.2.1 KioskNet

KioskNet is envisioned as a means of providing low-cost access to the Internet in remote developing regions [10]. Implemented as a set of tightly coupled software modules written in Java, C/C++, Perl and shell scripts running on Linux, KioskNet forms the underlying network infrastructure for my implementation. (Infrastructure components, as described in Section 3.3, run Debian whereas kiosk terminals run Ubuntu.) The system is almost entirely developed by a team of faculty members, graduate students, part-time undergraduate research assistants and summer interns at the University of Waterloo [40].

Because there is currently no known efficient routing mechanism for DTNs, messages sent and received within KioskNet are flooded throughout the network. So, for example, the SellProduce message generated by Karina in the kiosk usage scenario outline in Section 3.6 would be placed on every mobile router that connects to Karina's kiosk controller and all subsequent infrastructure components these routers encounter until the message reaches its destination and an acknowledgement sent by the wholesaler is successfully received by Karina. The process of sending and receiving messages (and acknowledgements) over KioskNet is implemented by OCMP (Opportunistic Connection Management Protocol) [34] via an API identical to the Secure Directory API described earlier in Section 5.2.2.

### 7.2.2 Cryptography Libraries

The two cryptography libraries used in my implementation are OpenSSL [23] and BouncyCastle [41]. The former provides a C API via its accompanying `libcrypto` library while the latter provides a Java API. Both libraries are open-source, updated regularly and used extensively by the open-source community.

## 7.3 Entity Credentials

As briefly described in Section 5.2.1, Entity Credentials consist of signing and encryption 2048-bit RSA public/private key pairs and corresponding X.509 certificates binding the public portions of these key pairs to the owner's identity. Signing key pairs are typically long-lived with their lifespans measured in months, whereas encryption key pairs are typically short-lived, with their lifespans being measured in weeks. In a rural public computing network, each entity, be it a user, franchisee, franchiser, ASP, kiosk controller or infrastructure component, is assigned a unique identifier called an *Endpoint ID* (EID). (An example of an EID for a kiosk user is `sumair.kiosk.village.franchisee.franchiser`.) As can be seen from the above example, EIDs are hierarchical in nature, with the top-level entity being the regional franchiser. X.509 certificates held by entities follow the same structure, with each certificate being signed by the authorizing entity one level above it in the hierarchy of entities. The only exception to this hierarchical structure is handheld users: handhelds and their users share the same EID, key pairs and certificates because it is assumed that each handheld will only have one user.

## 7.4 Rural Kiosks

In this section, I describe the implementation of my security architecture for rural Internet kiosks, detailing my implementation of digitally signed software updates, authenticated remote shell commands, tamper-evident system logs, the Secure Directory API and encrypted home directories for users.

### 7.4.1 Digitally signed Software Updates

Digitally signed software updates are produced and applied by authorized franchiser administrative personnel. To authenticate these updates, administrative personnel sign them with a designated private signing key, part of an Entity Credential as described earlier in Section 5.2.1, attaching the corresponding public key certificate signed by the franchiser. Signatures on software updates are automatically verified by infrastructure components before the update is applied, with the signing certificate chain attached to each update being verified with the franchiser's public signing key, which is installed when an infrastructure component is first set up. This same mechanism may be used to update the franchiser's public signing key, if necessary.

### 7.4.2 Authenticated Remote Shell Commands

Authenticated remote shell commands are produced and distributed in an identical fashion to signed software updates, as described above. As with software updates, infrastructure components verify the signatures on remote shell commands before running them.

### 7.4.3 Tamper-evident System Logs

Operational logs produced by infrastructure components contain debugging information, error reports, notifications of software updates, records of shell commands executed by the administrator, and fingerprints of all executables present on the devices. These logs are periodically sent to administrative personnel, allowing them to monitor deployed infrastructure components.

Logs are secured using a combination of hash chains, MACs (Message Authentication Codes), and a unique, randomly-generated symmetric key, $K_{Log_0}$, which is installed in the `/root` directory of an infrastructure component when it is set up. $K_{Log}$'s are used to produce a MAC for 24 hours of log entries at the end of each day before these log entries are rotated and combined with previous logs. In cryptographic terms, this is

$$MAC(Log_i) = MAC(\, K_{Log_i}, Log_i \,||\, i\,),$$

where $i \geq 1$, $Log_i$ is today's log entries, $K_{Log_i} = H(K_{Log_{i-1}})$, $H$ is a cryptographic hash function and $K_{Log_0}$ is the initial $K_{Log}$. Logs can be verified by validating the corresponding MACs. With this scheme, any attempt by an attacker to remove or modify a day's log can be detected. For non-repudiation purposes, each set of logs is signed with the source device's private key and the corresponding certificate is attached.

I note that this scheme does not protect logs against modification during the day if the above hash-chaining approach is applied to logs collected at the end of each day. This is reasonable if it is assumed that attacks on logs will only take place after they are sent to the franchiser (but before they are received). To provide increased protection, logs could be collected more frequently, with the most secure solution being to apply this scheme to all logs collected just before they are sent to franchiser personnel.

### 7.4.4 Secure Directory API

As described earlier in Section 5.2.2, the Secure Directory API is implemented as a daemon process which "watches" a configurable set of directories for files. This daemon is thus an atypical file server, implementing a file-based API similar to that available on Bell Labs' Plan 9 platform [26]. The Secure Directory API daemon is implemented as a multi-threaded Java application that makes use of the open-source BouncyCastle Java Cryptography package [41] for cryptographic operations.

For an example of how the Secure Directory API is used, on a kiosk terminal applications simply write outgoing data to a user's `~/application/supload` directory and read incoming data from the `~/application/sdownload` directory, where `~/` corresponds to the mount point on the terminal for the user's encrypted home directory (described earlier in Section 5.2.2). Applications writing outgoing files (e.g., `filename.ext`), must also write an associated metadata file called `filename.ext.desc` in the same `~/application/supload` directory. This metadata file contains a destination tag of the form `DEST: <destination-eid>`, where `destination-eid` is the end-point ID (EID) of the recipient (e.g., `sumair.kiosk.villagename.franchisee.franchiser`), and an optional certificate tag of the form `ATTACH_CERT` which tells the daemon to attach the sender's complete X509 certificate chain.

When new files are detected in an application's outgoing `~/application/supload` directory the daemon is configured to service, the daemon first reads the corresponding metadata file to determine the intended recipient. It then searches through the local White Pages database to obtain the recipient's public encryption key. If this key is not found or the X509 certificate containing it is found to no longer be valid, the outgoing file is ignored. If the key is found, the daemon first uses the sender's private signing key to digitally sign the outgoing file, attaching this signature to the file to create a message *M*. The daemon then generates a random 256-bit *Message Encryption Key* (MEK) using a cryptographically-secure Pseudo-Random Number Generator (PRNG) (included in the BouncyCastle package), and then encrypts the outgoing message M using this MEK with the AES-256 symmetric cipher in Cipher-Block Chaining (CBC) mode. The MEK is then encrypted with the recipient's public encryption key. The result is a signed, encrypted message, *EM*, of the form:

$$EM = E_{MEK}(M \,||\, S_{Sender_{Signing_{Pri}}}(M) \,||\, SigningCertificate_{Sender}) \,||\, E_{Recipient_{Encryption_{Pub}}}(MEK)$$

If the optional `ATTACH_CERT` tag is set in the outgoing file's metadata file, the encrypted message includes the sender's X509 encryption key certificate, producing an *EM* of the following form:

$$EM = E_{MEK}(M \,||\, S_{Sender_{Signing_{Pri}}}(M) \,||\, SigningCertificate_{Sender}) \,||\, E_{Recipient_{Encryption_{Pub}}}(MEK) \\ ||\, EncryptionCertificate_{Sender}$$

When new files are detected in an application's incoming `~/application/download` directory (note that this is different from `~/application/sdownload`), the daemon first checks the EM to see if a sender's encryption certificate was attached. If this certificate is present, the daemon attempts to validate it. If this certificate is found to be invalid or the sender's EID encoded in it does not match that of the incoming message, the daemon stops processing the incoming message and discards the message, reporting an error in its log. If not, the daemon parses the encryption certificate to obtain the sender's public encryption key. If no sender's certificate is attached to the message, the daemon obtains the sender's EID from the incoming message and searches through its local cache of the White Pages database to find the sender's encryption certificate. If no such certificate is found or the certificate found is no longer valid, the message is discarded and an error logged. (The sender is not a valid user.) The daemon uses the recipient's private decryption key to decrypt the incoming EM to reveal the plaintext message M, signature and signing certificate. This signing certificate is then verified, failure of which results in the message being discarded. If the sender's public signing key in this signing certificate can be used to successfully verify the message signature, the validated plaintext message is copied into the application's incoming `~/application/sdownload` directory for processing by the application.

I note that initial messages produced by a new user (messages sent before the user appears in the White Pages directory) include the new user's complete encryption certificate chain, allowing his/her messages to be processed even if the recipient's White Pages database has not yet been updated to reflect the new user. This is only allowed for a limited perion of time measured from the "valid-after" field in the encryption certificate (e.g., the maximum amount of time required for a White Pages update to propogate to a kiosk).

### 7.4.5 Encrypted User Home Directories

All user data, such as a user's pictures and emails, is stored in kiosk controllers and exported over NFS for access via terminals connected to a kiosk controller. As described in Chapter 5, to protect user data stored in kiosk controllers, users' home directories are created in encrypted virtual volumes. A user's virtual volume is exported in their encrypted form to terminals over NFS for automatic mounting and decryption when the user logs in using a mount extension to the Linux Pluggable-Authentication Module (PAM). The process is reversed when the user logs out. The use of an on-demand block device encryption scheme (DM_Crypt) ensures that a user's entire virtual volume does not need to be decrypted when a user logs in, and encrypted when the user logs out.

Blocks in the virtual volume are decrypted and encrypted only when they are read or modified, minimizing the impact of these operations on the performance of terminals.

Virtual volumes are encrypted using the AES-256 symmetric cipher on a randomly-generated key, $K_{UserHomeDir}$. This key is then replicated, one copy is encrypted with the franchiser's public key to produce $K_{UserHomeDir_{Backup}}$ and stored in the kiosk controller's `/root` directory and another copy is encrypted with a key derived from the user's login password to produce $K_{User}$ such that $K_{User} = E_{Password}\,[\,K_{UserHomeDir}\,]$.

The encrypted key $K_{User}$ is stored on the kiosk controller, alongside the user's encrypted virtual volume in the `/home` directory. (The `/home` directory on the kiosk controller is exported over NFS to terminals.) When a user logs in, the terminal first automatically decrypts $K_{User}$ using the user's password to obtain $K_{UserHomeDir}$ and then uses this key to decrypt the user's encrypted virtual volume.

In the event users forget their passwords, they can contact the local franchiser and ask to have their passwords reset. The franchiser, using the Secure Remote Shell, can then reset the user's password and create a new $K_{User}$ by using the franchiser's private key to decrypt the appropriate $K_{UserHomeDir_{Backup}}$ key retrieved from the controller's `/root` directory. When a user runs out of space in his/her home directory, we create a new, larger volume keyed with the same $K_{UserHomeDir}$ and sync the two volumes. I note that alternative schemes, such as those used by telecom service providers to replace users' lost SIM cards via telecom franchisees could be used instead of the scheme described above.

## 7.5 Infrastructure

The implementation of my security architecture on infrastructure components is similar to that on rural kiosks, as described above. All infrastructure components (i.e., mobile routers, kiosk gateways and kiosk proxy servers) feature digitally signed software updates, authenticated remote shell commands and tamper-evident system logs, as described in Section 7.4. Kiosk gateways and proxy servers additionally make use of the Secure Directory API for secure communication over the rural public computing network with kiosk controllers and mobile devices.

## 7.6 Issues Encountered

In this section, I briefly discuss some of the more interesting issues I encountered while implementing my security architecture.

### 7.6.1 Public/Private Key pairs

Contrary to the description in Section 7.3, entities in my implementation only possess a single public/private key pair, which in effect means they decrypt and sign data with the same private key – a potential security risk. This implementation decision affects entities in my model differently: (a) kiosk controllers, mobile routers and gateways only use their private keys to sign data, (b) proxy servers use their private keys for both signing and decryption, with both of these operations being implemented by the Secure Directory API (described in Section 7.4.4) which only produces signatures on hashed input data, (c) kiosk users, handheld users and ASPs use their private keys for both signing and decryption also through the Secure Directory API under the same conditions described earlier and (d), both franchisers and franchisees only use their private keys for signing data. Initially, this decision was made to reduce bandwidth usage costs in an implementation of my security architecture for the KioskNet platform [10], which uses an SMS (Short Message Service) data channel over cellular networks to complete user registration between kiosk controllers and proxy servers [22]. Public key certificates corresponding to signing keys could be attached to outgoing messages, however, with only the certificates corresponding to users' encryption keys being placed in the White Pages database. This approach would increase message sizes but has the advantage of being consistent with the current security architecture. This change (the use of two key pairs) will be made in the production version of my implementation.

### 7.6.2 Encrypted User Home Directories

In Section 7.4.5, I describe how I implement per-user encrypted home directories. As part of this implementation, I had to choose between either (a) mounting users' encrypted virtual volumes at the kiosk controller and exporting these via NFS, over an SSH tunnel, to kiosk terminals or (b), exporting the encrypted virtual volumes via NFS, without using an SSH tunnel, for mounting and decryption at kiosk terminals. My decision was to go with the latter option, as this reduces the computational load on kiosk controllers, distributing it instead to the recycled PCs serving as kiosk terminals. Chapter 8 includes a performance analysis of encrypted home directories implemented in this manner, demonstrating the feasibility of this approach.

### 7.6.3 White Pages on Kiosk Terminals

Because the encryption of outgoing application messages produced at kiosk terminals requires the public keys of recipients, it is necessary to provide access to the White

Pages database on kiosk terminals. Implementation options in this case were to either (a) place the White Pages database on the kiosk controller and provide kiosks terminals with read-only access to it over an SSH tunnel or (b), replicate the White Pages database on each kiosk terminal when a user logs in. Despite the additional load on kiosk controllers, the former option was selected to ensure kiosk terminals always have access to the most recent White Pages records available on the kiosk controller. (The White Pages database may be updated after a user logs into a kiosk terminal.)

# Chapter 8

# Performance Evaluation

In this chapter, I present a performance evaluation of my implementation with respect to its impact on users' experience and the increased load on kiosks and infrastructure components due to the added computational requirements imposed by my security implementation.

## 8.1 Typical Hardware setup

My experimental setup consists of a 1.2GHz x86-based system with 1GB of RAM and a 5400rpm 40GB hard disk as a kiosk controller, connected via 100Mbps Ethernet to a 1.8Ghz Pentium IV-based system with 1GB of RAM as a kiosk terminal.

## 8.2 Effect on User Experience

I now present an analysis of the impact of my security architecture on the performance of a rural Internet kiosk with the hardware setup described above.

### 8.2.1 Creating New User Accounts

As described earlier in Section 5.2.2, user accounts are created on kiosk controllers, with users' home directories placed in encrypted virtual volumes. These virtual volumes are created as disk images that must be initialized from one of either `/dev/zero` or `/dev/urandom`, with the latter providing more security as encrypted users' files written to their virtual volumes are harder to distinguish from pseudo-random data than from simple zeros. The performance cost of these two alternatives is shown in Figures 8.1 (log time scale) and 8.2, with the creation of a new user account of size 1GB taking a little over 20mins with pseudo-random data and approximately 45s with zeros. Experiments were repeated 3 times for the indicated file sizes using the test setup described earlier in Section 8.1.

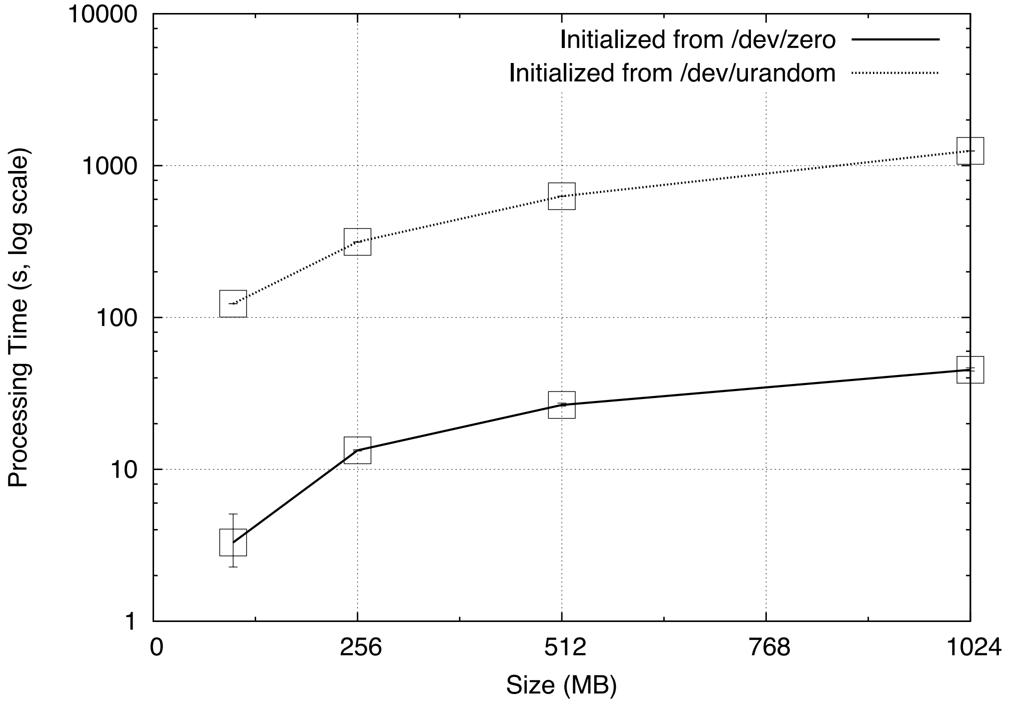

Figure 8.1: Creating Virtual Volumes (log time scale)

Using `/dev/urandom` to initialize users's virtual volumes guards against some cryptanalytic attacks, particularly where the attacker has physical access to the hard drive these volumes are stored on. So, although our test setup makes use of `/dev/zero` to mimimize the overhead of creating new user accounts, because users' virtual volumes are exported over NFS by kiosk controllers, it necessary to initialize these volumes from pseudorandom data for improved security.

In order to improve performance, an alternative approach to using `/dev/urandom` proposed by one of my readers, Prof. Ian Goldberg, is to initialize users' virtual vol-

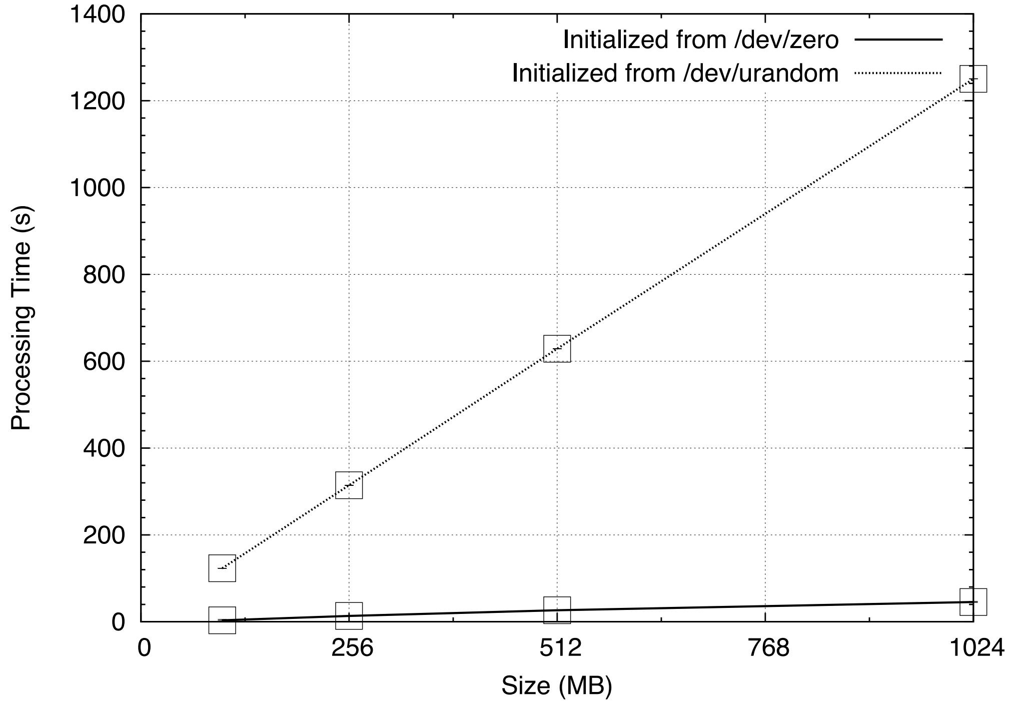

Figure 8.2: Creating Virtual Volumes (linear time scale)

umes with pseudorandom data generated by an RC4 stream cipher fed by `/dev/zero`. Implementation of this optimization is left as future work.

### 8.2.2 Logging In/Out of Terminals

Typical login times without encrypted home directories were in the range of 0.4s, with corresponding logout times of around 1.4s. Enabling encrypted home directories added approximately 100ms to both these times, an additional latency which I believe would likely go unnoticed. Experiments to measure login and logout delays were repeated 3 times using the test setup described earlier in Section 8.1 with a single user account that had been allocated 1GB of storage space as a virtual volume.

### 8.2.3 Sending/Receiving Secure User Data

Figure 8.3 reveals the performance of the Secure Directory API, which as outlined earlier in Section 5.2.2 is used by users' applications to securely send and receive data over KioskNet. Secure outgoing data is first signed and then encrypted using the RSA-2048 and AES-256-CBC ciphers. Secure incoming data is decrypted, the sender's public key certificate chain verified, and the corresponding signature on the data verified.

With the exception of the anomaly in Figure 8.3 for file sizes of 500KB (this warrants further investigation), I note that processing times increase linearly with file sizes and that a user would have to wait 560ms for a 1MB message to be signed and encrypted and 750ms for the same message to be decrypted and authenticated. These differences in processing times can be attributed to differences in the times required to encrypt/decrypt and sign/verify with RSA, as well as the additional time required to validate the sender's public key certificate chain attached to incoming messages.

As with previous experiments, experiments used to produce the dataset for Figure 8.3 were repeated 3 times for the indicated file sizes using the test setup described earlier in Section 8.1.

### 8.2.4 Reading/Writing to Home Directories

Figures 8.4 and 8.5 show data rates for reading and writing to users' home directories over NFS both with and without encryption. Experimental data reveals a drop in performance when reading files over 800MB in size without encryption, with a similar drop appearing for files over 250MB in size with encryption. I believe this occurs when

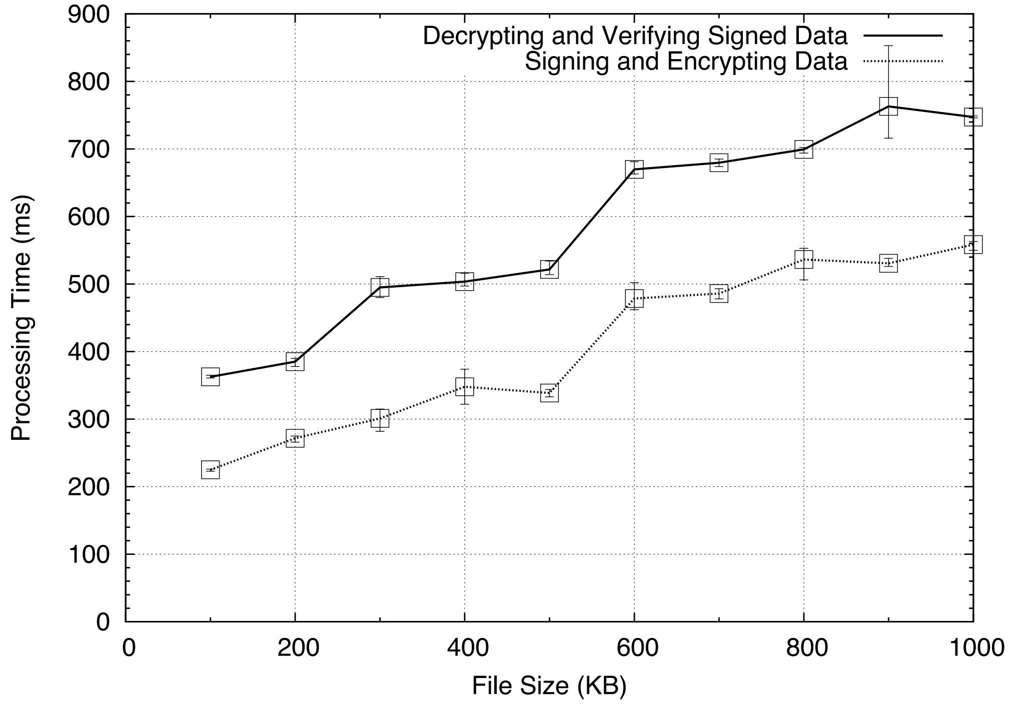

Figure 8.3: Performance of the Secure Directory API

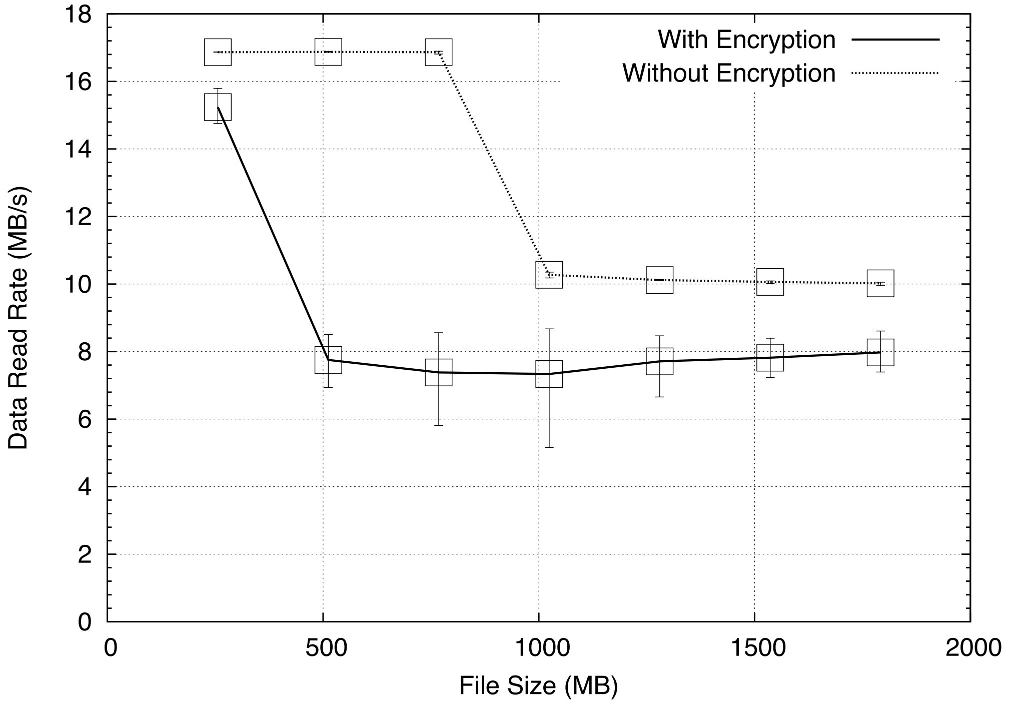

Figure 8.4: Reading User Data

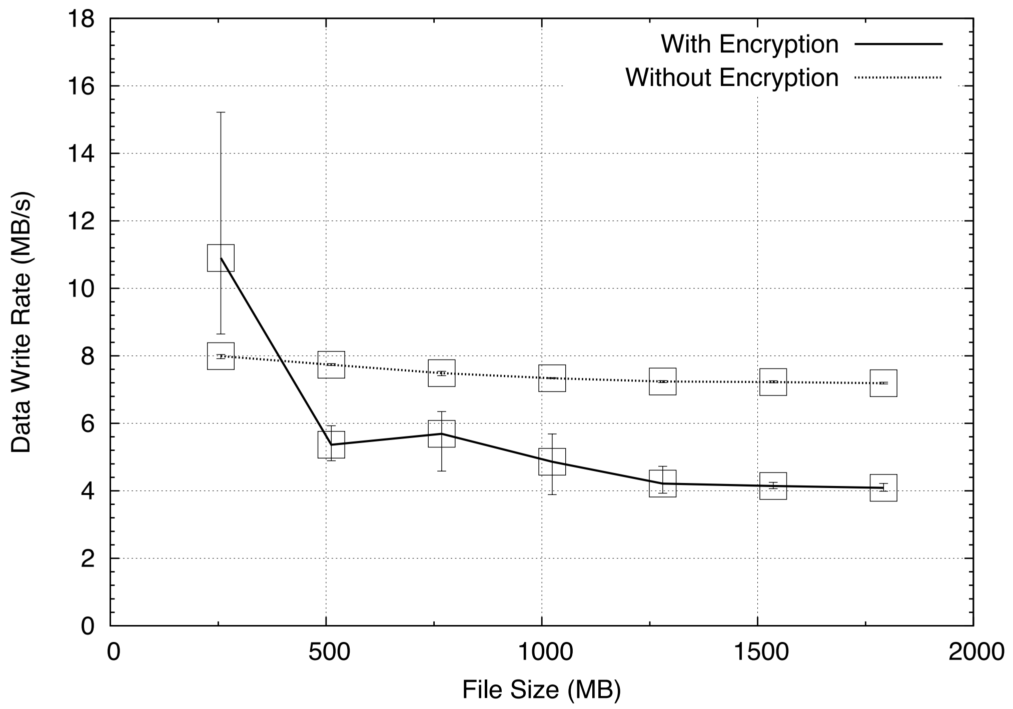

Figure 8.5: Writing User Data

the terminal's cache maintained in its RAM disk fills and the system begins paging out to its NFS-mounted swap on the kiosk controller. Further investigation is required to determine why this performance drop appears earlier when encryption is used. (I note that in a production system, this NFS-mounted swap space would also have to be encrypted, further reducing performance.)

As expected, average data read/write rates with encryption are lower than without encryption, with users experiencing drops in average read and write rates of approximately 2MB/s [1] and 3MB/s, respectively, with typical read and write rates over NFS without encryption averaging around 10MB/s and 7MB/s, respectively.

As with previous experiments, experiments used to produce the dataset for Figures 8.4 and 8.5 were repeated 3 times for the indicated file sizes using the test setup described earlier in Section 8.1.

## 8.3 Load on Infrastructure Components

Because no additional security-related operations are performed on mobile routers and gateways as a result of my security architecture, there is no effect on the performance of these systems. The proxy server, however, is required to function as an end-point in secure communication between users and legacy servers on the Internet, placing additional load on this system.

[1]MB/s = 1 Megabyte (1,048,576 bytes) per second

# Chapter 9

# Conclusion

In this Chapter, I conclude this thesis by summarizing my contributions and briefly discussing some potential directions for future work.

## 9.1 Summary and Contributions

In this thesis, I have presented a practical, unobtrusive and easy-to-use security architecture for rural public computing thats uses a combination of physical and cryptographic mechanisms to protect user data, public computing infrastructure and handheld devices that access this infrastructure. Key contributions of this work include (a) a detailed threat analysis of such systems with a particular focus on rural Internet kiosks and handheld devices, (b) a security architecture for rural public computing infrastructure that does not require any specialized hardware, (c) an application-independent security API for securely sending and receiving data between these systems and the Internet that can operate over periodically disconnected links, (d) an implementation of my scheme for rural Internet kiosks and (e) a performance evaluation of this implementation to demonstrate its feasibility.

## 9.2 Future Work

Potential directions for future work include a feature-complete implementation and evaluation of my security architecture for handhelds, further strengthening of my security architecture to account for untrusted franchisers, a practical technique to establish trust in kiosk terminals which does not require any specialized hardware and a third-party review of my implementation and use of cryptographic mechanisms.

# Appendices

# Appendix A

# Sample X509 Certificates

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 5 (0x5)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=CA, ST=Ontario, L=Waterloo, O=KioskNet, OU=KioskNet Franchisee, CN=Waterloo
        Validity
            Not Before: Jun 26 00:09:35 2008 GMT
            Not After : Jun 26 00:09:35 2009 GMT
        Subject: C=CA, ST=Ontario, L=Waterloo, O=KioskNet, OU=KioskNet User, CN=Kiosk Administrator
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b2:9c:ec:94:2e:70:55:aa:a5:4d:14:b6:02:4d:
                    71:0e:fb:fd:9e:9f:43:a8:fb:64:59:b1:68:41:ea:
                    ab:48:fa:83:56:39:80:68:35:02:76:b6:cc:c8:45:
                    21:28:65:7a:9b:e2:d4:aa:d4:2f:4a:95:cd:97:25:
                    05:c7:a6:01:d9:f1:7b:2c:dd:7c:84:14:6f:b9:ca:
                    e4:21:09:7d:07:59:1b:79:d0:3b:e8:2c:d2:44:71:
                    bf:80:75:12:de:5c:d1:2d:ec:77:2a:d7:6c:85:4d:
                    18:14:66:8d:6c:e1:20:5b:b6:be:11:a3:7f:be:9e:
                    85:83:f7:5b:66:ce:15:2b:30:ce:ed:f9:c2:47:b3:
                    9a:d3:73:7c:6f:59:22:3b:40:f8:2b:4a:1e:54:d6:
                    fb:da:a4:0d:8a:2a:ab:63:51:0e:44:51:f6:3f:5f:
                    2c:1a:84:4a:69:55:a4:41:0e:56:5b:17:cf:6a:6d:
                    67:0d:c3:6d:b3:77:30:c1:69:dd:f9:7f:c1:40:36:
                    e8:e1:ae:b7:23:29:38:3d:15:50:a0:18:de:85:45:
                    21:20:ce:08:82:61:ba:65:65:c3:c5:d4:fc:c8:76:
                    dc:53:e9:a9:1e:e6:73:eb:33:ad:60:3a:1a:37:1c:
                    f7:7b:96:ee:c2:dd:93:4d:e9:89:d6:b2:43:72:27:
                    9c:2b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                "OpenSSL Generated Certificate"
            X509v3 Subject Key Identifier:
                98:C6:54:27:56:98:B9:61:48:2A:F5:B3:3C:6F:C8:8D:5F:61:21:86
            X509v3 Authority Key Identifier:
                keyid:8B:08:9C:3C:4B:2D:69:7A:A1:91:25:1C:84:E6:62:46:00:F9:57:01
                DirName:/C=CA/ST=Ontario/L=Waterloo/O=KioskNet/OU=KioskNet Franchiser/CN=UW Test
                serial:02

            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:admin.kiosk.tetherless.uw.kiosknet.org
    Signature Algorithm: sha1WithRSAEncryption
        0f:9c:a3:a8:0e:d2:3a:e1:c8:37:ec:36:cb:d2:c2:1f:99:56:
        68:6c:b6:12:80:4c:90:12:46:9c:cf:79:fa:8e:3e:86:cb:f4:
        fb:d4:04:48:fe:7a:be:c7:b2:af:a2:b3:0a:f3:5e:58:a6:e0:
        6f:02:6f:8f:ef:fd:f2:3c:6e:23:90:f4:f0:f5:41:b4:b1:70:
        65:36:5e:f9:05:93:b7:ac:08:48:88:d5:f7:2a:89:20:98:fa:
        6a:7f:e8:60:1c:72:77:06:10:19:58:e2:1a:6a:89:96:7c:b5:
        02:28:6d:90:0e:32:7e:25:0b:49:87:84:66:4b:77:c7:5a:1d:
        48:9d:37:3e:15:3c:9a:23:cb:4a:a2:40:a4:15:2b:0d:28:b1:
        c3:9a:ff:8d:f5:2a:3b:98:ee:9f:52:3f:d9:fd:06:39:bf:43:
        f7:e8:a1:92:9c:0e:0d:0c:0b:ad:06:5b:f5:4a:b5:41:47:3f:
        1f:51:6d:2e:fe:61:35:86:a4:bd:d2:c0:27:dc:92:6a:ee:4f:
        7a:35:9d:29:07:83:62:86:a2:32:a5:79:c3:9d:5b:33:ce:b4:
        a8:d9:93:09:09:d7:c7:48:31:37:df:e9:a3:f4:00:75:5b:bb:
        cb:29:16:7e:3a:bc:37:8f:0e:60:3d:be:4e:1d:79:21:84:10:
        03:9a:ae:87
-----BEGIN CERTIFICATE-----
MIIErjCCA5agAwIBAgIBBTANBgkqhkiG9w0BAQUFADB2MQswCQYDVQQGEwJDQTEQ
MA4GA1UECBMHT250YXJpbzERMA8GA1UEBxMIV2F0ZXJsb28xETAPBgNVBAoTCEtp
b3NrTmV0MRwwGgYDVQQLExNLaW9za05ldCBGcmFuY2hpc2VlMREwDwYDVQQDEwhX
YXRlcmxvbzAeFw0wODA2MjYwMDA5MzVaFw0wOTA2MjYwMDA5MzVaMHsxCzAJBgNV
BAYTAkNBMRAwDgYDVQQIEwdPbnRhcmlvMREwDwYDVQQHEwhXYXRlcmxvbzERMA8G
A1UEChMIS2lvc2tOZXQxFjAUBgNVBAsTDUtpb3NrTmV0IFVzZXIxHDAaBgNVBAMT
E0tpb3NrIEFkbWluaXN0cmF0b3IwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQCynOyULnBVqqVNFLYCTXEO+/2en0Oo+2RZsWhB6qtI+oNWOYBoNQJ2tszI
RSEoZXqb4tSq1C9Klc2XJQXHpgHZ8Xss3XyEFG+5yuQhCX0HWRt50DvoLNJEcb+A
dRLeXNEt7Hcq12yFTRgUZo1s4SBbtr4Ro3++noWD91tmzhUrMM7t+cJHs5rTc3xv
WSI7QPgrSh5U1vvapA2KKqtjUQ5EUfY/XywahEppVaRBDlZbF89qbWcNw22zdzDB
ad35f8FANujhrrcjKTg9FVCgGN6FRSEgzgiCYbplZcPF1PzIdtxT6ake5nPrM61g
Oho3HPd7lu7C3ZNN6YnWskNyJ5wrAgMBAAGjggFAMIIBPDAJBgNVHRMEAjAAMC4G
CWCGSAGG+EIBDQQhFh8iT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUiMB0G
A1UdDgQWBBSYxlQnVpi5YUgq9bM8b8iNX2EhhjCBnwYDVR0jBIGXMIGUgBSLCJw8
Sy1peqGRJRyE5mJGAPlXAaF5pHcwdTELMAkGA1UEBhMCQ0ExEDAOBgNVBAgTB09u
dGFyaW8xETAPBgNVBAcTCFdhdGVybG9vMREwDwYDVQQKEwhLaW9za05ldDEcMBoG
A1UECxMTS2lvc2tOZXQgRnJhbmNoaXNlcjEQMA4GA1UEAxMHVVcgVGVzdIIBAjAL
BgNVHQ8EBAMCBeAwMQYDVR0RBCowKIImYWRtaW4ua2lvc2sudGV0aGVybGVzcy51
dy5raW9za25ldC5vcmcwDQYJKoZIhvcNAQEFBQADggEBAA+co6gO0jrhyDfsNsvS
wh+ZVmhsthKATJASRpzPefqOPobL9PvUBEj+er7Hsq+iswrzXlim4G8Cb4/v/fI8
biOQ9PD1QbSxcGU2XvkFk7esCEiI1fcqiSCY+mp/6GAccncGEBlY4hpqiZZ8tQIo
bZAOMn4lC0mHhGZLd8daHUidNz4VPJojy0qiQKQVKw0oscOa/431KjuY7p9SP9n9
Bjm/Q/fooZKcDg0MC60GW/VKtUFHPx9RbS7+YTWGpL3SwCfckmruT3o1nSkHg2KG
ojKlecOdWzPOtKjZkwkJ18dIMTff6aP0AHVbu8spFn46vDePDmA9vk4deSGEEAOa
roc=
-----END CERTIFICATE-----
```

Figure A.1: X509 Certificate for Kiosk Administrator

# References

[1] ISO/IEC 18004-2006: QR Code 2005 Bar Code Symbology Specification. *International Organization for Standardization (ISO), Geneva, Switzerland*, August 2006. 31

[2] BlackBerry Enterprise Solution: Security Technical Overview. Technical report, Research In Motion Limited, 2008. 7, 10

[3] N. Asokan, K. Kostianinen, P. Ginzboorg, J. Ott, and C Luo. Towards Securing Disruption-Tolerant Networking. Technical Report NRC-TR-2007-007, Nokia Research Center, March 2007. 6

[4] D. Balfanz and E. W. Felten. Hand-Held Computers Can Be Better Smart Cards. In *Proc. of 8th USENIX Security Symposium*, August 1999. 4

[5] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In *Proc. of the 8th Information Security Conference (ISC 2005)*, September 2005. 7, 9

[6] D. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. Rivest. The Untrusted Computer Problem and Camera-Based Authentication. In *Proc. of Int'l Conference on Pervasive Computing (Pervasive 2002)*, pages 114–124, August 2002. 6

[7] M. D. Corner and B. D. Noble. Zero-Interaction Authentication. In *Proc. of the 8th annual international conference on Mobile computing and Networking*, pages 1–11, 2002. 7, 8

[8] S. Farrell, S. Symington, H. Weiss, and P. Lovell. Delay-Tolerant Networking Security Overview - draft-irtf-dtnrg-sec-overview-04. Internet Draft, February 2008. 6

[9] S. Garriss, R. Cáceres, S. Berger, R. Sailer, L. van Doorn, and Z. Zhang. Towards Trustworthy Kiosk Computing. In *Proc. of 8th IEEE Workshop on Mobile Computing Systems and Applications (HotMobile'07)*, February 2007. 4, 31, 40, 42

[10] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, U. Ismail, and S. Keshav. KioskNet: A System for Low-Cost Internet Access For Developing Regions. Proc. of ICTD, December 2007. 1, 44, 50

[11] P. Gutmann. Plug-and-Play PKI: A PKI your Mother can Use. In *Proc. of 12th USENIX Security Symposium*, pages 45–58, August 2003. 26, 34

[12] ICICI Bank's Microfinance Strategy: A Big Bank thinks Small. `http://www.microfinancegateway.org/content/article/detail/13446`. Accessed August 2008. 1

[13] A. Kate, G. Zaverucha, and U. Hengartner. Anonymity and Security in Delay Tolerant Networks. In *Proc. of 3rd Int'l Conference on Security and Privacy in Communication Networks (SecureComm 2007)*, September 2007. 1, 6, 7

[14] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol - RFC 2401. Internet Standard, November 1998. 7

[15] I. Krstíc and S. L. Garfinkel. Bitfrost: the One Laptop per Child Security Model. In *Proc. of 3rd Symposium on Usable Privacy and Security (SOUPS 2007)*, pages 132–142, July 2007. 7

[16] R. Kumar and M. L. Best. Impact and Sustainability of E-Government Services in Developing Countries: Lessons Learned from Tamil Nadu, India. *The Information Society*, 22(1):1–12, June 2006. 1

[17] R. Luk, M. Ho, and P. M. Aoki. Asynchronous Remote Medical Consultation for Ghana. In *Proc. of the 26th ACM SIGCHI CHI Conference (CHI 2008)*, April 2008. 1

[18] M. Lukac, L. Girod, and D. Estrin. Disruption Tolerant Shell. In *Proc. of 2006 ACM SigComm Workshop on Challenged Networks (CHANTS 2006)*, September 2006. 29

[19] P. MacKenzie and M. K. Reiter. Delegation of cryptographic servers for capture-resilient devices. In *Proc. of the 8th ACM conference on Computer and Communications Security*, pages 10–19, 2001. 7, 8

[20] P. Mackenzie and M. K. Reiter. Networked cryptographic devices reilient to capture. *International Journal of Information Security*, 2(1):1–20, November 2003. 7, 8, 9

[21] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using Camera phones for human-verifiable authentication. In *Proc. of 2005 IEEE Symposium on Security and Privacy*, May 2005. 33

[22] Earl Oliver. Exploiting the short message service as a control channel in challenged network environments. In *Proc. of the third ACM MobiCom Workshop on Challenged Networks (CHANTS 2008)*, September 2008. 50

[23] OpenSSL. `http://www.openssl.org/`. Accessed June 2008. 44

[24] A. Oprea, D. Balfanz, G. Durfee, and D. K. Smetters. Securing a Remote Terminal Application with a Mobile Trusted Device. In *Proc. of 20th Annual Computer Security Applications Conference (ACSAC 2004)*, pages 438–447, December 2004. 6

[25] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *IEEE Computer*, 37(1), January 2004. 1

[26] R. Pike, S. Presotto, D. Dorward, B. Flandrena, K. Thompson, H. Trickey, and P. Winterbottom. Plan 9 from Bell Labs. *Computing Systems*, 8(3):221–254, 1995. 27, 47

[27] RIM BlackBerry. `http://www.blackberry.com`. Accessed June 2008. 7, 9

[28] W. Scheirer and M. Chuah. DTN Security Features Technical Report. Technical report, Department of Computer Science and Engineering, Lehigh University, April 2006. 7

[29] D. Scott, R. Sharp, A. Madhavapeddy, and E. Upton. Using visual tags to bypass Bluetooth device discovery. *Mobile Computing and Communications Review*, 1(2), January 2005. 33

[30] K. Scott and S. Burleigh. Bundle Protocol Specification - draft-irtf-dtnrg-bundle-spec-03. Internet Draft, January 2006. 7

[31] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems. In *Proc. of 20th ACM Symposium on Operating Systems Principles (SOSP 2005)*, pages 1–15, October 2005. 4, 5

[32] A. Seth and S. Keshav. Practical Security for Disconnected Nodes. In *Proc. of 1st Workshop on Secure Network Protocols (NPSec 2005)*, pages 31–36, 2005. 1, 6, 7

[33] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost Communication for Rural Internet Kiosks Using Mechanical Backhaul. In *Proc. of 12th International Conference on Mobile Computing and Networking (MOBICOM 2006)*, pages 334–345, September 2006. 1

[34] A. Seth, S. S. Bhattacharyya, and S. S. Keshav. Application Support for Opportunistic Communication on Multiple Wireless Networks. *Manuscript*, 2005. 44

[35] R. Sharp, J. Scott, and A. R. Beresford. Secure Mobile Computing via Public Terminals. In *Proc. of 4th Int'l Conference on Pervasive Computing (Pervasive 2006)*, pages 238–253, May 2006. 6

[36] D. K. Smetters, D. Balfanz, G. Durfee, T. Smith, and K. Lee. Instant Matchmaking: Simple and Secure Integrated Ubiquitous Computing Environments. In *Proc. of 8th International Conference on Ubiquitous Computing (UbiComp 2006)*, September 2006. 7, 9

[37] A. Surie, A. Perrig, M. Satyanarayanan, and D. Farber. Rapid Trust Establishment for Transient Use of Unmanaged Hardware. Technical Report CMU-CS-06-176, School of Computer Science, Carnegie Mellon University, December 2006. 4, 5, 40

[38] S. Symington, S. Farrell, H. Weiss, and P. Lovell. Bundle Security Protocol Specification - draft-irtf-dtnrg-bundle-security-05. Internet Draft, February 2008. 6, 7

[39] Telecentre.org. `http://www.telecentre.org`. Accessed June 2008. 1

[40] Tetherless Computing Group. `http://blizzard.cs.uwaterloo.ca/tetherless`. Accessed June 2008. 44

[41] The Legion of the Bouncy Castle. `http://www.bouncycastle.org/`. Accessed June 2008. 44, 47

[42] Trusted Computing Group. `https://www.trustedcomputinggroup.org`. Accessed August 2008. 5

[43] United Villages. `http://www.unitedvillages.com`. Accessed June 2008. 1

[44] Voltage Security, Inc. `http://www.voltage.com`. Accessed June 2008. 7