Securing KioskNet: A Systems Approach

Sumair Ur Rahman, Urs Hengartner, Usman Ismail and S. Keshav
David R. Cheriton School of Computer Science,
University of Waterloo, Waterloo ON N2L 3G1, Canada
{surrahman, uhengart, uismail, keshav}@cs.uwaterloo.ca

Technical Report No. CS-2007-43

ABSTRACT

Internet kiosks typically provide weak security guarantees and therefore cannot support secure web access or transaction-oriented applications such as banking and bill payment. We present the design and implementation of a practical, unobtrusive and easy-to-use security architecture for KioskNet, a system for low-cost rural Internet kiosks. Our system uses a combination of physical and cryptographic mechanisms to protect user data and infrastructure nodes despite frequent disconnections, untrusted administrators, and large end-to-end delays. Our contributions include (a) a detailed threat analysis of rural kiosks, (b) the first comprehensive security architecture for rural kiosk security, (c) a simple and easy to use API for securely sending and receiving files over KioskNet (d) protection for user data both on the file system and in transit, and (e) a usable public key infrastructure (PKI). Performance measurements show that our system imposes 100ms of delay during user login at kiosks and an additional 560ms and 750ms per megabyte of data sent and received by a kiosk user, respectively, suggesting that the system is both efficient and usable.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection; C.2.1 [**Network Architecture and Design**]: Store and forward networks, Wire-less communication

General Terms

Design, Measurement, Security

Keywords

Delay tolerant networks, kiosk security, public key infrastructures, system design

1. INTRODUCTION

Envisioned as a means of providing low-cost Internet connectivity to under-served rural communities, KioskNet is a delay-tolerant network of purpose-built servers, lightweight embedded systems and recycled PCs that cooperate to provide end users with transaction-oriented offline WWW access through rural kiosks [5].

The potential deployment of KioskNet over a large geographic region, with the majority of the system components operating in remote rural areas and largely unsupervised, creates an interesting set of security challenges. In this paper, we address these challenges by considering the interests of all stake holders. We propose a practical, unobtrusive and easy-to-use security architecture that uses a combination of physical and cryptographic mechanisms to protect KioskNet, its users and its operators.

The key contributions of this work include (a) a detailed analysis of the threats faced by KioskNet, (b) the first comprehensive security architecture for rural kiosk security and an implementation of this architecture, (c) a simple and easy to use API for securely sending and receiving files over KioskNet, (d) protection for user data both on the file system and in transit, and (e) a usable public key infrastructure (PKI) based on off-the-shelf building blocks. An evaluation of our system's impact on KioskNet's performance reveals an additional 100ms delay during user login at kiosks and an additional 560ms and 750ms per megabyte of data sent and received by kiosk users, respectively.

The rest of this paper is organized as follows. In section 2, we present our system model, followed by a comprehensive threat analysis in section 3. We continue in section 4 by presenting our security architecture. Section 5 analyses our security architecture and its effectiveness in preventing or mitigating the attacks presented in section 3. Before reviewing related work in section 8, we discuss implementation issues in section 6 and evaluate the performance of our implementation in section 7. We then conclude and briefly discuss potential directions for future work in section 9.

2. SYSTEM MODEL

In this section, we introduce KioskNet, describe the entities that operate, support or use the system, and then outline two typical usage scenarios.

2.1 Overview

Each KioskNet deployment provides service to users in a specific geographic region and is independently administered by a *Franchiser* (see section 2.2). We illustrate a typical deployment scenario in figure 1. KioskNet components 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 network connectivity by means of WiFi, GPRS, SMS, VSAT, or dial-up.

WWW2008, April 21–25, 2008, Beijing, China.
.

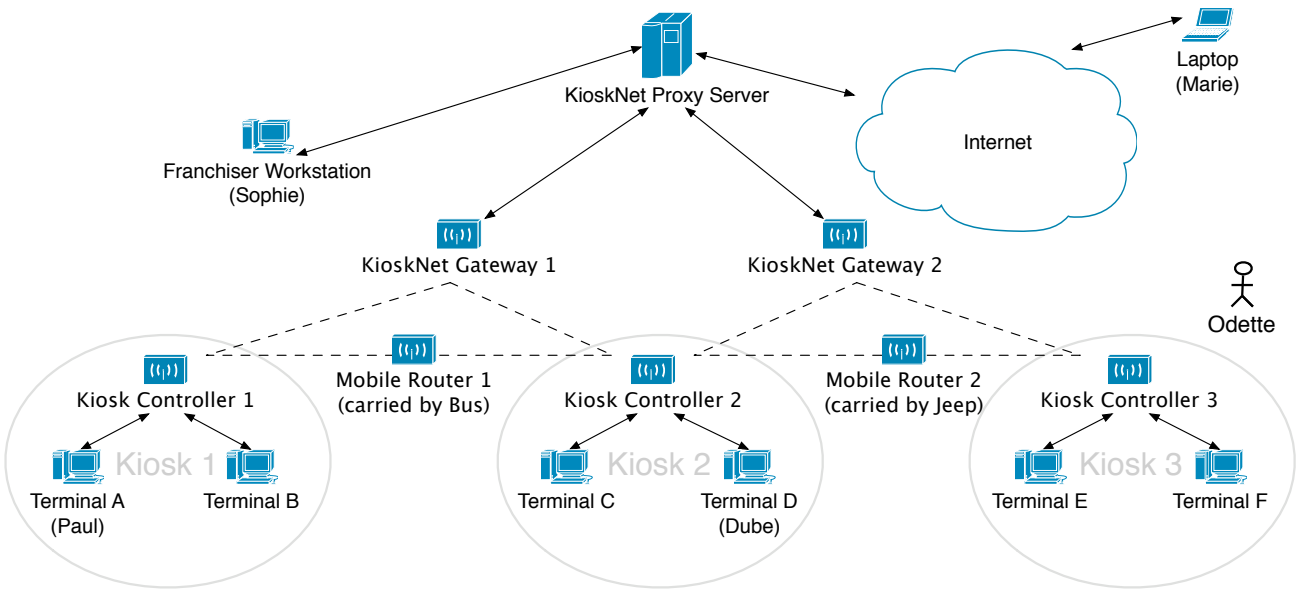


Figure 1: Typical KioskNet deployment scenario

- *Kiosk Terminals* – inexpensive recycled PCs that are capable of running Linux and that allow users to connect to the regional KioskNet based on a wired connection to a single kiosk controller.
- *KioskNet 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 KioskNet gateways and rural kiosks. Mobile routers opportunistically communicate with gateways and kiosk controllers to transport data between them.
- *KioskNet Proxies* – servers deployed at data centers that serve as a proxy between KioskNet gateways and disconnection-unaware legacy servers on the Internet.

We define a *Kiosk* as a set of one or more *Terminals* and a single *Kiosk Controller*. The servers and lightweight embedded systems that form the backbone of a KioskNet, namely kiosk controllers (but not terminals), gateways, mobile routers and proxies, are termed *Infrastructure Components*. Infrastructure components are owned and operated by the single franchiser responsible for a particular region. For a more detailed discussion, we refer to Guo et al. [5].

2.2 Entities

The following entities have an interest in the correct and reliable operation of a KioskNet deployment:

- *Franchisers* – franchisers are public or private organizations that own, operate and administer KioskNet infrastructure components deployed in a particular geographic area.
- *Franchisees* – franchisees are private organizations or individuals licensed by a KioskNet franchiser to operate terminals connected to a kiosk controller provided by the franchiser.
- *Transport Providers* – transport providers are public or private organizations that provide road or rail transportation within geographic areas serviced by a KioskNet deployment. Transport providers are contracted by franchisers to carry mobile routers on their vehicles to enable mechanical backhaul between gateways and kiosk controllers.
- *Application Service Providers (ASPs)* – application service providers are public or private organizations that are licensed by franchisers to deploy their applications on a KioskNet. For example, a bank might be an ASP using KioskNet to provide micro finance and banking services to farmers.
- *Users* – users subscribe to KioskNet services with a franchiser, usually through a franchisee, to access services, such as applications provided by ASPs, and the Internet using terminals owned and operated by local franchisees.

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. We exploit this organizational structure in section 4 when proposing the use of a public key infrastructure (PKI) in our security architecture.

2.3 Usage Scenario A: Email

In this usage scenario, we describe how Paul, a KioskNet user, sends an email to two recipients, Dube, a user registered in the same KioskNet region, and Marie, who is not a KioskNet user and must be reached via the Internet.

1. Paul logs into a terminal at his local kiosk and uses an email client to prepare an email addressed to Dube (`dube@region.kiosknet.org`) and Marie (`marie@someisp.net`). When the message is ready to be sent, Paul's email client transfers the email to an IMAP server on the same terminal, which in turn sends it to a KioskNet email server running on the kiosk controller connected to the terminal.
2. The email server generates two sets of *bundles* from the email, the first set addressed to Dube and the second set to Marie.
3. Mobile routers carried by vehicles passing by Paul's kiosk destined for a nearby city opportunistically pick up the bundles and deliver them to a KioskNet gateway in the city.
4. The gateway schedules the set of bundles destined for Dube to be forwarded back into the KioskNet via the next available mobile router and forwards the set of bundles destined for Marie to the KioskNet proxy server over the Internet.
5. The proxy server receives the set of bundles destined for Marie, unpacks it, and forwards the email to Marie's POP server over the Internet. Meanwhile, a vehicle carrying a mobile router destined for the rural kiosk servicing Dube picks up his set of bundles.
6. The set of bundles containing Paul's email to Dube is delivered to the kiosk controller serving Dube's rural kiosk, unpacked, and stored in his account's mailbox to be read when he next logs into a terminal.

2.4 Usage Scenario B: Remote Maintenance

In this usage scenario, we describe how an authorized franchiser administrator, Sophie, and a field technician employed by the franchiser, Odette, generate and apply a set of software upgrades, termed a *maintenance package* for the purpose of this example, to kiosk controllers both remotely over KioskNet and physically by means of a USB memory stick inserted into the devices.

1. As part of regular maintenance activities, Sophie prepares a maintenance package to be applied to all kiosk controllers operated by her employer using a purpose-built KioskNet maintenance application. The maintenance package is flooded into the KioskNet via the franchiser's proxy server and a copy is given to Odette on a USB memory stick.
2. A maintenance server running on the franchiser's proxy server receives the maintenance package, breaks it up into bundles, and forwards these to all gateways in the franchiser's KioskNet.
3. Vehicles carrying mobile routers destined for rural kiosks in the franchiser's region that pass by its gateways opportunistically pick up the maintenance package bundles for subsequent delivery to its kiosk controllers.
4. Kiosk controllers receive the maintenance package bundles, unpack and assemble them to obtain the maintenance package, and then trigger a maintenance client, which verifies the package before applying it.
5. During routine maintenance checks at rural kiosks served by her employer, Odette discovers a kiosk controller that has yet to receive Sophie's maintenance package. She inserts the USB memory stick containing the maintenance package into the kiosk controller, whose maintenance client then detects the package and verifies it before applying it.

3. THREAT ANALYSIS

In this section, we identify potential attackers, describe their capabilities, and list key attacks against KioskNet.

3.1 Potential Attackers

In addition to outsiders, defined as being none of the concerned entities introduced in section 2.2, we assume attackers to be any of the following three entities: franchisees, transport providers or users. KioskNet 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. We exclude ASPs under the premise that any software, data and configuration changes by these entities will be inspected and approved by the appropriate franchiser before being introduced into the franchiser's KioskNet.

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, 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 administrator passwords for these systems.
- *Technical Expertise* – the experience and technical expertise required to modify the software or configuration of a Linux-based system, given sufficient network-based or physical access.
- *Knowledge of KioskNet* – an understanding of the architecture and operational characteristics of KioskNet as an open-source software system.

3.2 Recognized Threats

Threats against KioskNet can be categorized as attacks against the confidentiality, integrity or availability of the system. In terms of confidentiality, we are concerned with the privacy of user data and any secret keys stored in their accounts (see section 4.2 for details). In terms of integrity, we are concerned with the integrity of this data plus the integrity of infrastructure components, terminals and the impersonation of franchiser personnel and kiosk users. For availability, we consider the jamming of wireless links between infrastructure components. (A terminal is connected to a controller by a wired link.) When combined with potential attackers, these threats give us the grid in figure 2 below. We also classify each attack according to its likelihood.

The classification of likely and unlikely threat-attacker combinations in figure 2 is based on the capabilities of a

Figure 2: Recognized Threats against KioskNet vs. Potential Attackers

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 the device's wireless interface. Similarly, although an outsider may have the motivation and ability to view, fabricate, or modify data carried by a mobile router, a transport provider carrying the device would be more likely to launch such an attack given easier physical access.

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.

4. SECURITY ARCHITECTURE

In this section, we highlight our security goals with respect to the concerned entities introduced in section 2.2 and then go on to describe how our security scheme protects users, the terminals they use, and all KioskNet infrastructure components.

4.1 Security Goals

Because it is impossible to completely secure a complex distributed system such as KioskNet, our 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, such as TPMs (Trusted Platform Modules) [15] or a modifiable BIOS.

Specific security goals, in terms of the four entities that use or operate KioskNet, are as follows: Franchisers are concerned with the security of their infrastructure and want to detect, if not prevent, the misuse of their infrastructure components by any of the concerned entities. Franchisees are concerned with the security of their kiosk and want protection against the spread of viruses over and any attacks launched through KioskNet. Depending on the type of service they provide, ASPs may want franchisers to guarantee the integrity of their software when deployed on a KioskNet. Examples of such software might include tax payment and land registry systems operated by the government. Finally, users are only concerned with the confidentiality and integrity of their data. We detail the mechanisms used to achieve these goals in the following subsections.

4.2 User and Operator Security

KioskNet entities that use or operate the system each possess a unique set of *Entity Credentials*. Entity Credentials consist of an RSA key pair and a corresponding X.509 certificate that binds the holder's identity to the public part of its key pair.

KioskNet users and operators obtain and use their Entity Credentials as described below:

- *KioskNet Franchisers* – franchisers self-generate an RSA key pair and then use the public part of this key pair to obtain a certificate signed by the KioskNet root CA based in our lab at the University of Waterloo in Ontario, Canada. This key pair is then used to sign certificates issued to franchiser administrative personnel, licensed franchisees, and ASPs. Franchiser personnel may in turn use their credentials to authenticate secure updates and remote shell commands as described in section 4.3.1.

- *KioskNet Franchisees* – franchisees obtain certificates in a similar fashion to franchisers, with the only difference being their certificates are signed by their franchiser. Franchisees use their key pairs to sign certificates issued to users registered at their kiosks.

- *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.

- *KioskNet Users* – users obtain their credentials when they register at a rural kiosk. Their certificates are signed by the local franchisee. The usage of certificates and key pairs is transparent to users, which is important since previous research has shown that users cannot be expected to manually deal with certificates [6]. Namely, a key pair and a certificate are automatically created upon registration and stored in the user's encrypted home directory (see section 4.4.1). Furthermore, usage of the keys is simplified through the *Secure Directory API* (described in section 4.4.2), where incoming data is transparently decrypted and verified, and outgoing data is transparently encrypted and signed without user intervention.

  Certificates for users are made available to other KioskNet users, franchisers, ASPs, and other franchisees by means of a KioskNet user database termed the *White Pages*. This database is maintained by each region's franchiser and updates to it are periodically sent out 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.

All certificates described above are chained to the KioskNet root CA's certificate 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 environments.

4.3 KioskNet Infrastructure Security

Infrastructure components are protected against attacks through the use of a combination of cryptographic and physical security mechanisms.

In terms of physical security, we assume that infrastructure components, specifically kiosk controllers, mobile routers and KioskNet 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 we 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. Similar to Entity Credentials, Infrastructure Credentials consist of a key pair generated by the device and a 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. A device uses its credentials to sign its operational logs, 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 or operators of mobile routers.

4.3.1 Secure Software Updates

Secure software updates are produced by authorized franchiser administrative personnel and applied as outlined earlier in section 2.4. To authenticate these updates, administrative personnel sign them with a designated private key, part of an Entity Credential. Infrastructure components hold the corresponding public key, which is installed when a component is first set up, and use this key to verify updates before applying them. If necessary, a secure software update can be used to update this key itself.

4.3.2 Secure Remote Shell

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

4.3.3 Secure Operational 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 flooded over the KioskNet for eventual delivery to franchiser 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

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

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.

4.4 Terminal Security

Terminals are PCs, typically recycled, that network boot from a read-only image stored in a kiosk controller. These images contain the terminal's kernel, configuration files and applications. Because terminals may be disk-less, 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 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 we describe in the following two subsections.

4.4.1 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 highlighted in section 3.2, 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. A user's virtual volumes are exported in their encrypted form to terminals over NFS for automatic mounting and decryption when the user logs in. The process is reversed when the user logs out. The use of an on-demand block device encryption scheme ensures that a user's entire virtual volume does not need to be decrypted when a user logs in, and vice-versa 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} or

$$K_{User} = E_{Password}\,[\, K_{UserHomeDir}\,].$$

The key K_{User} is stored alongside the user's virtual volume in `/home`. 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 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 simply create a new, larger volume keyed with the same $K_{UserHomeDir}$ and sync the two volumes.

In the future, with the increasing availability of fingerprint readers, in particular on laptops, biometrics-based login could give more security than password-based login. However, stealing fingerprint information will remain a possibility, though likely more difficult, and the usage of biometrics raises privacy concerns.

4.4.2 Secure Directory API

The *Secure Directory API* allows developers to easily produce applications that communicate securely over KioskNet. Applications can simply write outgoing data to a user's `/home/user/application/suploa`d directory and read incoming data from the `/home/user/application/sdownload` directory, where `/home/user` corresponds to the mount point on the terminal for the user's encrypted home directory (described above in section 4.4.1).

A daemon running on the terminal automatically decrypts and verifies signatures on incoming data using the user's private key (stored in his/her encrypted home directory) and other users' public keys available in the White Pages database (introduced in section 4.2) and places it in the `sdownload` directory. Similarly, this daemon automatically encrypts and signs all outgoing data placed in `suploa`d after looking up the recipient's public key in the White Pages database. We note that data destined for a server reachable over the Internet is either encrypted on a specific ASP's public key or on the proxy's public key and then sent in plaintext over the Internet.

5. SECURITY ANALYSIS

In this section we present a brief analysis of the security mechanisms used to protect KioskNet. Figure 3 summarizes the security mechanisms employed by KioskNet and shows how these are combined to guard against the attacks presented earlier in section 3. Let us describe next how we defend against attacks on a kiosk by its users, its franchisee, and outsiders.

5.1 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 rebooted when a user logs out. The reboot kills any processes that the user might have left behind and that could use up CPU or memory resources.

5.2 Franchisee

By entering his/her password into a terminal during the login process, a user gets access to his/her data and his/her private key, both of them encrypted with a key derived from the password. Therefore, a franchisee might be tempted to steal a user's password. For example, the franchisee could set up a fake login environment and fool a user into revealing his/her password. The only way to avoid this attack with very high guarantee is for the user to query the state and configuration of the terminal in front of which he/she is sitting, which would allow him/her to detect that the login environment is fake. However, existing mechanisms for giving such a guarantee [4] require the user to have a trusted computing device, which is not the case in KioskNet. Therefore, we cannot completely avoid this attack, we can only make it (overly) expensive to execute for a franchisee. Let us discuss our defenses in more detail.

A PC serving as a terminal could include a hard disk, which a franchisee (legitimately) uses for booting non-KioskNet environments. This enables a malicious franchisee to create an application that displays a fake login screen to KioskNet users. There are two ways to get around this attack: First, we can train users to enter a non-overridable keyboard combination, such as Ctrl-Alt-Delete, before entering their passwords, which will reveal the fake login screen. Second, if a user still enters his/her password, the application then has three options to proceed, all of them are likely going to cause suspicion with the user: It can display an error message, saying that there is a problem and that the user should come back later, it can repeatedly ask the user to re-enter his/her password, till the user just gives up, or it could have the PC boot into KioskNet, where the user enters her password one more time. Our protection is user education, by means of a poster that warns users never to enter a password multiple times and to notify the franchiser of any suspicious behaviour.

It is much more difficult for the franchisee to set up a fake login application directly in the KioskNet environment running on a terminal, since the franchisee does not have the administrator password for the environment and cannot install applications (see section 4.3).

A determined franchisee could download the open-source KioskNet software package from our website, insert a keylogger, and have a terminal run the modified package. This attack also involves setting up a fake controller that provides the modified software or changing the terminal to store and boot the modified software directly from its hard drive. In short, this attack is very involved.

Both the fake login attack and the keylogger attack could be avoided with the help of a device that is trusted by a user. For example, we could exploit Surie et al.'s approach [13] and have a user boot a terminal from a trusted USB memory stick before logging in. A minimal trusted OS on the stick then downloads the KioskNet image from the controller, verifies its integrity, and, if successful, boots KioskNet. There are two problems with this approach: First, expecting users to carry a USB memory stick with them might not be realistic in rural environments. One way to address this problem is to have the franchiser deposit a read-only USB memory stick that is difficult to clone without detection with the franchisee. Second, terminals need to be able to boot from a USB memory stick, and it must not be possible to turn this off in the BIOS. Both requirements might not hold for (current) recycled PCS. However, we are looking into this option for the next version of KioskNet.

Digital fingerprints of user passwords are stored in the root partition of the controller and downloaded by a terminal when it boots. Passwords are salted, so that a franchisee needs to execute an expensive brute-force search for password recovery. The root partition is not encrypted. If it were encrypted, a franchisee would have to enter the decryption key into a controller at boot time, which is difficult, considering the controller has neither a keyboard nor a display. Physical security mechanisms, as described in section 4.3, make it difficult for a franchisee to extract the hard drive from a controller and to tamper with the software and

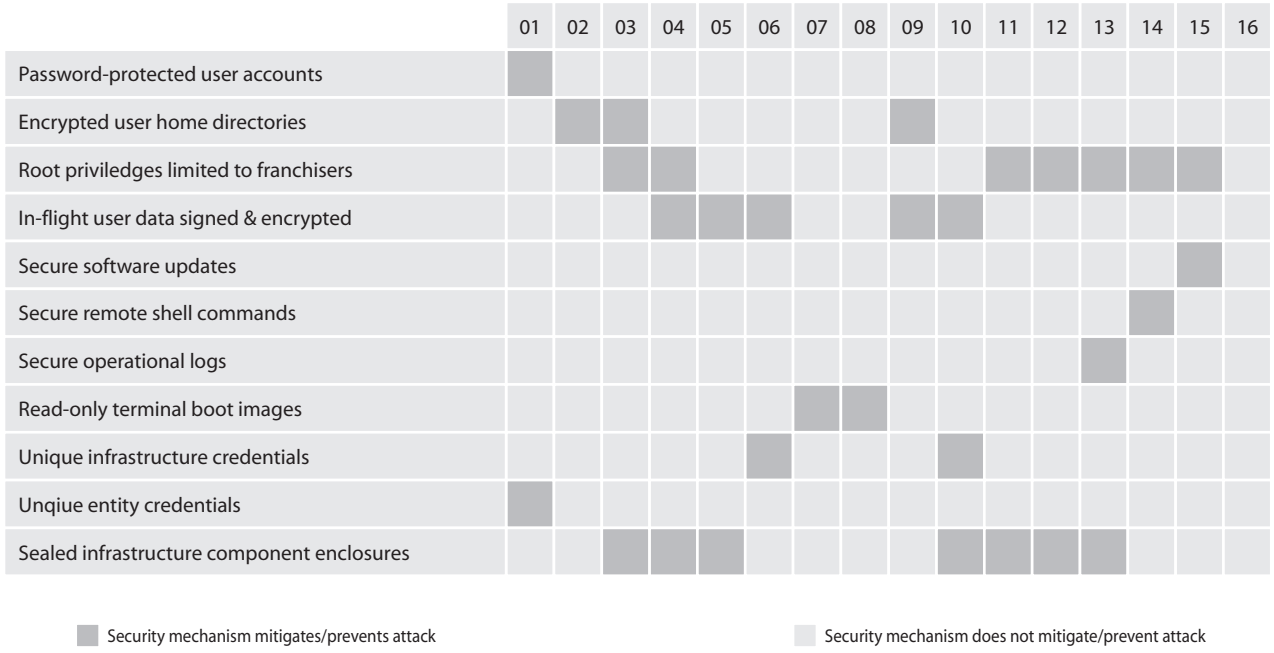


Figure 3: KioskNet Security Mechanisms vs. Recognized Threats

data stored on it (e.g., adding malicious software, replacing the certificate identifying the franchiser, or extracting the controller's private key).

Note that it is not possible to use authentication of a terminal by the controller as a security mechanism. Authentication requires the terminal to have some secret information. However, the terminal cannot store this information in a way that is not accessible to the franchisee. (As mentioned in section 4.1, we cannot assume the existence of TPM-based or BIOS-based mechanisms supporting safe storage of secrets.) Therefore, a franchisee could steal the secret information and use it to authenticate a fake terminal. Without authentication, the franchisee could access a user's data stored on the controller. However, the data is stored in encrypted form, which makes it useless to the franchisee.

5.3 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. IMPLEMENTATION ISSUES

The initial security architecture for KioskNet took advantage of Hierarchical Identity-Based Encryption (HIBE) [11], an extension to Identity-Based Encryption (IBE). This allows a kiosk user to send encrypted messages to another user without the need to know this user's public key [1]. Although academically interesting, using IBE turned out to be problematic in practice. The only IBE implementation ready for practical use is controlled by a single entity (Voltage Security, Inc.), which does not release source code and has stringent licensing conditions for commercial use. We therefore decided to replace IBE with our own PKI based on the `libcrypto` and `Bouncy Castle` open-source cryptographic libraries. With the help of the White Pages database, as explained in section 4.4.2, we ensure that recipients' public keys are always available to the sending kiosk.

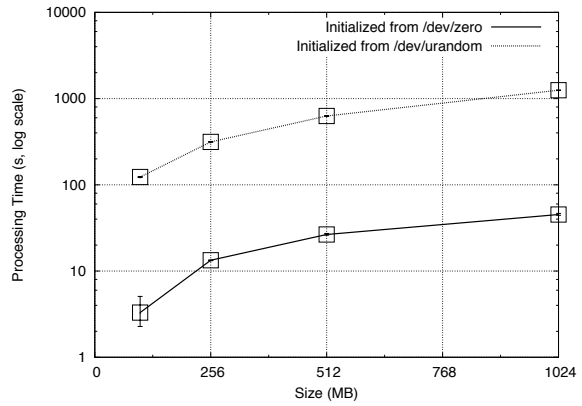
The implementation of our security architecture relied on a number of security add-ons for Linux including `openssl`, `cryptsetup`, `dmsetup` and `pam-mount` Debian packages, as well as the `aes` and `dm-crypt` kernel modules. All of these software components are free, open-source, and widely available over the Internet and through the `apt` package management systems bundled with both Debian and Ubuntu Linux.

7. PERFORMANCE EVALUATION

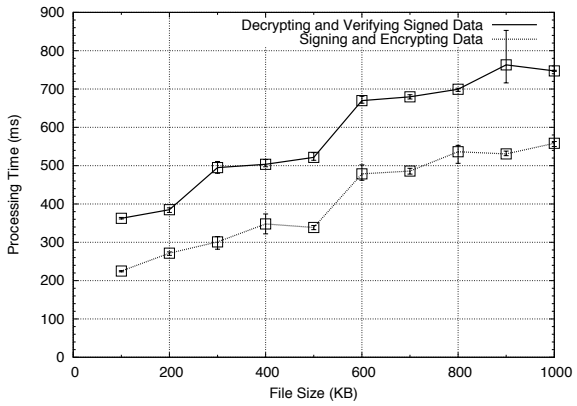
We now present an analysis of the impact of our security architecture on KioskNet's performance. Our experimental setup consists of a 1.2GHz x86-based system with 1GB of RAM and a 5400rpm 40GB hard disk as a kiosk controller, connected via 100Mbps Ethernet to a 1.8Ghz Pentium IV-based system with 1GB of RAM as a terminal.

7.1 Creating New User Accounts

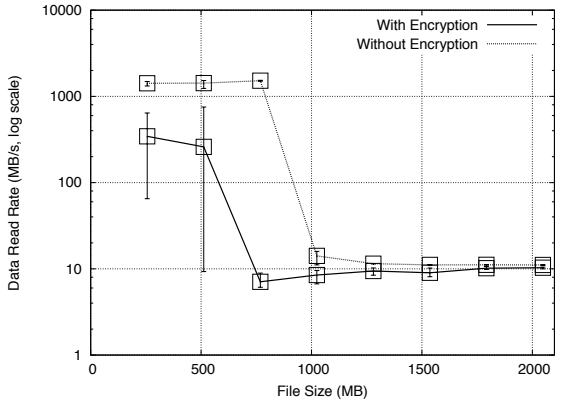
As described earlier in section 4.4.1, user accounts are created on kiosk controllers, with users' home directories placed in encrypted virtual volumes. These virtual volumes are cre-



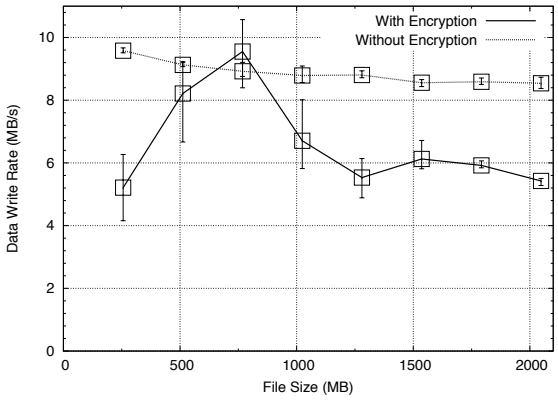
(a) Creating Virtual Volumes (note: log time scale)



(b) Secure Directory API



(c) Reading User Data (note: log rate scale)



(d) Writing User Data

Figure 4: Performance Evaluation

ated 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 figure 4(a) below, 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. Note that using `/dev/urandom` helps against some cryptanalytic attacks on a hard drive, where the attacker has physical access to the hard drive. In our threat model, this is not the case, so using `/dev/zero` is sufficient.

7.2 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 we believe would likely go unnoticed.

7.3 Sending/Receiving Secure User Data

Figure 4(b) below reveals the performance of the Secure Directory API, which as outlined earlier in section 4.4.2 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. We 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, where incoming data takes longer to process because of the additional time required to validate the sender's public key certificate chain.

7.4 Reading/Writing to Home Directories

Figures 4(c) and 4(d) above 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 for files over 800MB. We believe this occurs when 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. As expected, average data read/write rates with encryption are lower than without encryption for files of this size, with users seeing drops in average read and write rates of approximately 1MB/s and 3MB/s, respectively, with typical read/write rates over NFS without encryption averaging around 9MB/s in our setup.

8. RELATED WORK

Previous research has studied how users can access remote services via an untrusted proxy without revealing sensitive information, such as private keys or passwords, to the proxy. In some approaches [4, 10, 13], a user queries the proxy about its state and verifies that this state is trustworthy. These approaches require that the user has a trusted device to execute a query. Other approaches [2, 9, 12] offload the processing of sensitive information from the untrusted proxy to a user's trusted device. None of these approaches are applicable to our scenario, since we do not expect users to have a trusted device.

Bitfrost [8] is the security model of the "One Laptop per Child" project, whose goal is to distribute low-cost, networked laptops to children in the developing world. This way, the children will gather computer experience and can communicate with other children around the world. The project's goal is different from the goal in KioskNet, where people use shared kiosks, instead of individual laptops, to (potentially) perform secure transactions, such as online banking. The difference in the goals is also visible in the different security models. In Bitfrost, a user has full control over her laptop, and the security model makes it difficult, but not impossible, for the user (and software) to execute dangerous actions. In our model, users (and franchisees) have only limited control over a kiosk, and we need to defend against attacks trying to get additional control. The designers of Bitfrost 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. In Bitfrost, users create self-signed certificates that bind their name to a public key. In our security model, user certificates are signed by a third party. This way, a service, such as online banking, can, but does not have to, delegate identity verification to the third party, which can be important in rural environments. In both Bitfrost and KioskNet, the usage of certificates is transparent to users, and the infrastructure generates and processes certificates on their behalf.

There are several security architectures for delay-tolerant networks. Some architectures [1, 7, 11], two of them developed by co-authors of this paper, are based on identity-based cryptography. In section 6, we explain why we choose a PKI-based approach for KioskNet. The DTN research group also favors a PKI-based approach [3, 14]. The group explicitly considers key management an open issue, whereas we present a solution. Furthermore, the group concentrates on the secure exchange of messages between DTN nodes and the required format specifications, whereas our main concern is at the application and usability level, that is, how can DTN users create and receive secure messages in a usable way.

9. CONCLUSION AND FUTURE WORK

In this paper we have presented a comprehensive threat analysis for KioskNet, identified security goals for the system given the needs of a diverse group of stake holders, and proposed a practical, unobtrusive security architecture that meets these requirements. We have also provided a detailed security and performance analysis for an implementation of our proposed scheme.

Potential directions for future work include the use of smartcards and biometric authentication systems to provide end users with simplified, password-free access to kiosk terminals and support for the secure mobility of users and their data between kiosks.

10. REFERENCES

[1] 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.

[2] 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.

[3] S. Farrell, S. Symington, H. Weiss, and P. Lovell. Delay-Tolerant Networking Security Overview - draft-irtf-dtnrg-sec-overview-03. Internet Draft, July 2007.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] S. Symington, S. Farrell, H. Weiss, and P. Lovell. Bundle Security Protocol Specification - draft-irtf-dtnrg-bundle-security-04. Internet Draft, September 2007.

[15] Trusted Computing Group. `https://www.trustedcomputinggroup.org`. Accessed October 2007.