The Pennsylvania State University

The Graduate School

College of Engineering

PROTECT MY CHILD

A NEW PARADIGM IN PARENTAL CONTROL

A Thesis in

Computer Science and Engineering

by

Jonsimon Tack

©2016 Jonsimon Tack

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2016
The thesis of Jonsimon Tack was reviewed and approved* by the following:

Sencun Zhu  
Associate Professor of Computer Science and Engineering  
Thesis Advisor

Danfeng Zhang  
Assistant Professor of Computer Science and Engineering

Chitaranjan Das  
Distinguished Professor of Computer Science and Engineering  
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School
ABSTRACT

Parental control systems have existed for years and enjoy a sizable market. Recent advances in computing provide opportunities to apply new ideas to the parental control problem. Protect My Child (PMC) is a new paradigm in parental control systems. Targeted towards mobile devices, PMC relies on the notions of ubiquitous computing and high connectivity. PMC is fully mobile, allowing parents to control child devices from their own tablets or smartphones without the need for physical access to the child’s device. PMC provides intuitive, fine-grained, remote control of sensitive operations including text messaging, phone calls and app installs. By modeling the natural request-response paradigm that exists between parents and children, PMC facilitates in-band modification of access control policy. Child devices initially deny access to sensitive operations. When an operation is denied, the child is able, from his or her device, to request a policy change from the parents. The parents are immediately notified of the request on their own devices, where they may grant or deny access to the operation. The parental decision is sent back to the child device where it is stored and the access control policy is updated. This approach to access control avoids unintentional access to, or unintentional denial of, sensitive operations.

PMC mediates SMS text messages and phone calls by phone number, and app installation by app. The fine-grained control described above may be combined with coarse-grained controls as well. PMC mediation of phone calls and text messaging may be remotely enabled or disabled by the parents. For app installs, PMC offers three modes of mediation: request-each, age appropriate and allow-all. Request-each mode requires one-time explicit permission to install each app. Age appropriate mode implicitly allows installation of age appropriate apps based on the child’s age and app ratings, but requires one-time explicit permission to install each more restrictively rated app. Allow-all mode implicitly allows all app installs.

PMC also offers many industry standard parental control features, such as location tracking, emergency notifications, remote device locking, app locking and contact white/black lists. The design employs proven security principles to provide an effective, privacy-respecting parental control system. A prototype of the PMC system was implemented based on the Android Open Source Project and evaluated running on commodity devices. The PMC system is found to
mediate all sensitive operations with no unintentional denials and without unintentionally granting access to sensitive operations.
TABLE OF CONTENTS

List of Figures .............................................................................................................. v

List of Tables ................................................................................................................. vi

Acknowledgements ......................................................................................................... vii

Chapter 1 Introduction ................................................................................................. 1

Chapter 2 Background ................................................................................................. 5

Chapter 3 Related Work ............................................................................................... 8

Chapter 4 Problem Statement ..................................................................................... 10
  4.1 Limitations of Existing Approaches .................................................................... 10
  4.2 Desired Properties ............................................................................................... 12

Chapter 5 System Design ............................................................................................ 14
  5.1 Concepts ................................................................................................................ 14
  5.2 Assumptions .......................................................................................................... 15
  5.3 Functional Requirements ..................................................................................... 15
  5.4 System Architecture ............................................................................................ 20
  5.5 Security Model ...................................................................................................... 23
  5.6 Design Principles .................................................................................................. 23
  5.7 Usability ................................................................................................................ 25
  5.8 Security and Privacy ............................................................................................ 30

Chapter 6 System Implementation .............................................................................. 34
  6.1 Technologies .......................................................................................................... 34
  6.2 System Hooks ....................................................................................................... 34
  6.3 Reference Monitor ............................................................................................... 39
  6.4 Child App ............................................................................................................... 40
  6.5 Parent App ............................................................................................................ 40
LIST OF FIGURES

Figure 5-1. A Device Family .......................................................................................................... 15
Figure 5-2. System Architecture .................................................................................................. 20
Figure 5-3. Mediation of Operations ............................................................................................ 22
Figure 5-4. Device Communication ............................................................................................... 26
Figure 5-5. In-band Policy Management ....................................................................................... 27
Figure 5-6. Adding a Device to a Device Family ......................................................................... 28
Figure 6-1. Structure of a System Hook ....................................................................................... 35
Figure 6-2. SMS Send Hook in IccSmsInterfaceManager ............................................................. 35
Figure 6-3. SMS Receive Hook in InboundSmsHandler ................................................................. 36
Figure 6-4. Dial Call Hook in PhoneProxy .................................................................................... 36
Figure 6-5. Receive Call Hook in PhoneBase ................................................................................ 37
Figure 6-6. First App Install Hook in PackageManagerService ................................................... 38
Figure 6-7. Second App Install Hook in PackageManagerService ................................................. 38
Figure 6-8. App Uninstall Hook in PackageManagerService ........................................................ 39
Figure 6-9. Launch App Hook in ActivityManagerService .......................................................... 39
Figure 7-1. Creating a Device Family ........................................................................................... 44
Figure 7-2. Joining a Child Device to a Device Family (NFC) ...................................................... 45
Figure 7-3. Releasing a Device ..................................................................................................... 46
Figure 7-4. Local Denial of an SMS Send Operation ..................................................................... 47
Figure 7-5. Local Denial of a Place Call Operation .......................................................... 47
Figure 7-6. Local Denial of an App Install Operation .................................................... 48
Figure 7-7. In-band Policy Management ........................................................................... 49
Figure 7-8. SMS/Call White and Black Lists ................................................................. 50
Figure 7-9. Remote App Locking ..................................................................................... 50
Figure 7-10. Remote Device Lock and Unlock ............................................................... 51
Figure 7-11. Locked Child Device ................................................................................... 52
Figure 7-12. Child Lock Schedule ................................................................................. 52
Figure 7-13. Remote Location Tracking ......................................................................... 53
Figure 7-14. Sending an Emergency Notification ......................................................... 54
Figure 7-15. Remote App Uninstall ............................................................................... 55
LIST OF TABLES

Table 4-1. Desired System Properties ......................................................... 13

Table 5-1. Functional Requirements .............................................................. 16

Table 5-2. Sensitive Operations ................................................................. 17

Table 5-3. Message Server REST API ......................................................... 21

Table 5-4. System Design Properties, Challenges and Design Principles ........... 24

Table 5-5. System Usability Design Properties ............................................. 25

Table 5-6. System Usability Design Principles .............................................. 25

Table 5-7. App Mediation Modes ................................................................. 30

Table 5-8. Contact Mediation Modes ......................................................... 30

Table 5-9. Security and Privacy Properties ............................................... 31

Table 5-10. Security and Privacy Design Principles ..................................... 31

Table 5-11. Policy Change Capability Token Fields ..................................... 32

Table 5-12. Rights ................................................................................. 33

Table 5-13. Objects ................................................................................. 33
ACKNOWLEDGEMENTS

I gratefully acknowledge my adviser, Dr. Sencun Zhu, for his guidance over the course of this project and for providing mobile devices for development and testing. I thank Dr. Danfeng Zhang for serving on the committee. I thank and acknowledge Dr. Xin Chen, Dengfeng Li, Dr. Chu Huang and Steve Chen for their work on the design and implementation of various parts of the PMC prototype. Finally, I thank my family for their patience and support over the duration of this degree program.
Chapter 1

Introduction

Information systems are a source of benefits and risks for adults and children alike. For children however, the risks can be especially severe. Use of computer systems by juveniles carries with it dangers of inappropriate content exposure, bullying, online predators and even health problems. At the same time, the fact that children derive many educational and entertainment benefits from information systems is undeniable. The development of affordable, highly connected mobile devices has served to increase both the benefits and risks of information systems. Today’s smartphones and tablets are powerful, portable computer systems that connect to Wi-Fi, cellular data and voice networks. The portability and connectivity of these devices increases the risks to children. Mobile devices become highly personalized, privacy-sensitive personal items, exposing children to elevated risk due to tracking and device loss. The benefits of today’s mobile devices are even greater than those of classical information systems. Children can easily communicate with parents and guardians, have access to emergency services like 911, and enjoy social, educational and entertainment benefits with greater mobility.

Parents want to be able to provide these benefits to their children while mitigating the risks. Parental control system providers aim to facilitate this through easy-to-use and effective software. A parental control system is a type of system administration, where a parent specifies policies which are enforced on the child’s device. Such policies may limit the operations permitted by the device to remain within safe and healthy limits with respect to content and time of use. The management of policy has proven to be a difficult task even for trained professionals. This presents a challenge for parental control systems. These systems cannot assume a high degree of technical expertise on the part of the end users.

Industry has responded to the demand for parental control systems by providing a variety of options for parents. The Android marketplace, Google Play, offers many parental control and monitoring products from many vendors. While useful, these apps suffer from two critical flaws.
First, by design, third-party apps cannot restrict the behavior of other apps running on the same device. As a countermeasure against typical threats from malfunctioning apps and malware, the Android security model uses sandboxing to isolate apps from one another. As a result, existing parental control apps employ techniques such as customized app launchers or custom web browsers that replace the functionality provided by the mobile device software. Because such parental control apps are not actually privileged with respect to other apps, they can be bypassed using the device history or by terminating the parental control app from the task manager. Second, typical parental control systems employ policies that are based on coarse-grained controls, like web-site categories and app ratings. Such policies inevitably lead to content being unintentionally blocked. Because the device policy is based on high level abstractions, modifying the device policy becomes difficult for the parent. In the worst cases, parents temporarily or permanently abandon the parental control system so that the child can access benign content that is being blocked. This unintended blocking of content is due to the fact that the parent was not able to specify their intentions precisely enough in the form required by the parental control system policy. Some parental control systems also require the parent to have access to the child’s device to modify policy. Such a design is limited not only because the parent must be physically present to modify the device, but more importantly because the child’s device must allow policy modifications. Typically such controls are secured by a password-like authorization feature. If the password becomes known to the child, the parental control system policy can be easily modified or disabled.

The explosion of affordable mobile devices has ushered in an age of ubiquitous computing, and along with that, opportunities to apply new approaches to the parental control problem. A new opportunity exists to advance parental control systems in a way that was not possible before. Due to the connected nature of modern mobile devices, systems can be built where parents and children work together using their mobile devices to remotely administer child device policy in-band. Generally speaking, parents are always available via their mobile devices. This allows them to modify the policy of their children’s devices in near real time. Parental control systems can now be designed that model policy administration functionality according to the request-response relationship that exists between children and parents. As any parent knows, there is nothing more natural than a child asking for what they want, and a parent responding.
Protect My Child (PMC) is a new paradigm in parental control systems that leverages ubiquitous computing, high connectivity and proven software-security principles to provide a mobile parental control system that is easy to use, effective and flexible. The PMC system defines certain operations as sensitive operations. Such operations include SMS messages, phone calls, app install/uninstall, and general device usage. PMC employs classical security principles to enforce parental control policy on the child device. Specifically, PMC uses the reference monitor concept to enforce a capability-based policy on protected operations. The PMC reference monitor provides complete mediation, is isolated, and employs fail-safe defaults. This means that in the absence of a specific capability allowing the child access to a protected resource all such access attempts are denied. Capabilities to modify policy on child devices are generated and digitally signed by a parent’s device and verified on the child’s devices using public key cryptography. Once a capability has been provided for a specific operation, the child device policy is changed to allow the operation from that point on.

PMC avoids the need for parents to understand and administer complex policies by modeling a request-response paradigm that exists between parents and their children. Initially, the child device will deny all sensitive operations. Each time an operation is denied, PMC allows the child to request permission from the parent. This request is sent to the parent’s own device where the parent is immediately notified of the request and decides whether or not to allow access. If allowed, the parent device generates a digitally signed capability for the child device to perform the operation. The new capability is sent back to the child device where it is authenticated and applied, allowing the child to perform that operation in the future without the need to interact with the parent again. This approach to policy management is dynamic, fine grained, and simple. Children ask if they can perform a protected operation, say SMS with a new friend, and the parent says yes or no. In addition to responding to child permission requests, parents can push capabilities to the child’s device and revoke capabilities at will. Capabilities can be created to apply at certain times, or to expire. The parent does not need physical access to the child’s device in order to grant or revoke permissions nor does the parent need to be physically present with the child.
PMC provides three distinct advantages over current mobile parental control systems:

1. PMC is based on a simple, intuitive approach to policy management that is in-band, fine-grained, effective, and avoids unintentional denials.
2. PMC employs proven, classical security principles to provide a secure and privacy-respecting design.
3. PMC is fully mobile, allowing parents to control child devices from their own smartphones without the need for physical access to the child’s device.
Chapter 2

Background

Information systems provide many educational, social and entertainment benefits. They are also a source of many risks [1]. For children, the risks can be especially severe. Risks to juveniles include the dangers of inappropriate content exposure [2], bullying [3], online predators [4], and even health problems [5]. While parenting styles vary greatly, parents have authority and responsibility to supervise and protect their children [6]. While parental control systems are a tool available to help parents, ultimately it is the parents who decide whether or not to use them. Industry has responded to the demand for parental control software. In 2013, there was a $1 billion market for parental control systems [7]. In 2015, U.S. parental control app downloads topped 124 million [8].

Implicit in any control system, is the idea of policy. Policy defines the rights that a user has with respect to system objects. Parental control systems enforce policy on the controlled device. This policy is setup and managed by the parent. In general, policy management has turned out to be a challenge. This is true for example with SELinux, prompting research into tools to simplify the policy management task for students and professionals [9]. The more flexible a system policy is, the more difficult it becomes to administer. As a result, parental control systems often focus on coarse-grained policy controls that are simple to configure. However, coarse-grained policy results in both unintended access and unintended blocking of mediated operations or content. Fine-grained policy controls can prevent both accidental access and unintended blocking, but such systems can require large amounts of time to set up. Frequently parents must repeatedly tweak the access control policy. Historically, this problem has been exasperated by the fact that parents were not always available to make changes at the times desired by their children. Both parents and children become understandably frustrated with such systems. Often, parental control systems are disabled due to the combination of difficult policy management and undesired results.
With the wide availability of smartphones near the start of the 21st century, mobile computing arrived in a new way. Rather than carrying a laptop computer in a case, smart phones fit in the pocket. For many purposes, desktop personal computers and laptops have been supplanted as smartphones and tablets, running new operating systems, flooded the market. Mobile devices have become more than household items, they have become personal items. In 2015, The Pew Research center found that while smartphone and tablet ownership continues to increase, computer ownership is stagnant or in decline [10]. These mobile devices are highly connected, often providing multiple network technologies in a single device. Where Wi-Fi networks are unavailable, cellular data networks provide internet connectivity.

One of the major mobile device operating systems, Android, is maintained as a free and open source software project called the Android Open Source Project (AOSP). AOSP allows commercial companies, researchers, and hobbyists to develop customized versions of the Android operating system and run them on inexpensive developer-friendly mobile devices. Based on the Linux kernel, AOSP serves the research community in much the same way as Linux does. It is an excellent platform for education and experimentation and is also widely used in many commercial and critical systems.

While mobile devices are easily connected to data networks, they are not easily addressable. Because the devices change from one network to another so often, it is not practical to statically address these devices. This presents a challenge for distributed systems that involve mobile devices. Mobile devices can periodically check in with cloud-based servers, but this mechanism is costly in terms of battery usage. A cloud messaging service (CMS) provides a way to push messages to mobile devices regardless of location. Using a CMS eliminates the need to poll a message broker from the client. There are CMS providers available for both Android and iOS.

Mobile device security has been an area of active research for many years. Because these devices are essentially computers, there is a constantly evolving code base that exposes and patches security vulnerabilities. The most popular mobile device operating systems were designed with security principles in mind, such as app isolation and sandboxing. They also support encryption tools that can be used to provide classical security goals of confidentiality, integrity and availability. Rich cryptography libraries can be used to create privacy-respecting systems that do not share personal information with third parties.
The desire of parents to have usable parental control solutions persists, as is attested by the many options currently available on mobile app marketplaces such as the Google Play Store and the Apple App Store. The presence of ubiquitous, connected, mobile devices presents an opportunity to apply a new approach to the parental control problem where parents use their own mobile devices to remotely administer child devices *in-band*. In-band means that children can request policy changes from their parents from within the system and receive an immediate decision. This mechanism aligns with the request-response relationship that exists between parents and children. In the past, parents were typically unavailable to adjust access control policy when needed by the child, and the effort required to make adjustments was large. But today, thanks to modern mobile devices, parents are virtually always online and available to make in-band policy decisions. This places fine-grained control in a new light. The advantages offered – no unintended access or denials – still require many decisions to be made by parents. However, now those decisions can be made with minimal effort, even over great distances.
Chapter 3

Related Work

A great deal of commercial work has been done in the area of mobile parental control systems. There are mature tools available in app marketplaces such as the Google Play Store and the Apple App Store. These products can be categorized according to the types of capabilities they provide. Content filters block access to content considered undesirable by the parent. Usage controls limit device time or access to certain operations that the parent considers off limits. Monitoring tools record information about the activities of the child and report it to the parent in the form of alerts, reports, or both. Most parental control systems combine features from all three of these categories. Below we survey several of the most popular modern mobile parental control systems.

Norton Family is a richly featured parental control suite that runs on computers and mobile devices including Android and iOS devices. Norton Family offers content filtering, access controls and monitoring functionality. Norton Family requires users to create an account and provides a web interface for management of policy, and viewing reporting. The system is also sends email alerts to parents.

Net Nanny, available for PCs, Android and iOS, is a custom internet browser that replaces the system browser on the device and prevents other browser apps from launching. Net Nanny can also block other kinds of apps from running, including blocking all new apps by default. Policy is managed through a web interface. Net Nanny requires a user account and provides a web interface for reporting and policy management.

Qustodio is a content filtering, access control and monitoring app that works on computers and mobile devices. It is installed only on the child device. Monitoring and policy management is provided through a web portal. Qustodio requires a user account and provides daily reports to the parent by email.
Mobicip Safe Browser is a content filter for browsing the internet. It replaces the default browser on the child’s mobile device. Monitoring and policy management is provided by a web portal. A companion app, Mobicip Monitor allows the parent to remotely administer the policy and view reports.

MMGuardian is a parental control system that provides content filtering, monitoring, and access control that includes web filtering, text monitoring, app blocking, remote locking and time limit enforcement. Policy is managed from a web portal or a companion app. MMGuardian can be configured to send SMS messages to alert parents when a new app is installed.

Screen Time Parental Control is an app that lets parents control the amount of time their children spend on their mobile devices. Parents manage child devices from a web browser or their own devices. SMS messages can be sent alerting parents to new app installs.

Kids Place is a restricted launcher app for Android that a parent can configure on their own device. Kids Place is designed for sharing the parent’s device with the child. While the child is using the device, they are restricted to the apps that were selected by the parent ahead of time. A pass code is required to exit the Kids Place launcher and return the phone to normal. Kids Zone is another parental control app based on a restricted launcher for sharing a device with a child. Both Kids Place and Kids Zone also block access to in-app ads.
Chapter 4

Problem Statement

4.1 Limitations of Existing Approaches

*Parenting is a highly individualized activity.* What is acceptable to one parent may not be acceptable to others, and what a particular parent finds acceptable for one child may be different than what is acceptable for another. The variability in parenting suggests that parental control systems should be fine-grained. However, fine-grained control typically leads to complicated policy specification. Policy specification is a notoriously difficult task, even for those trained in computer security. When it comes to apps, existing approaches tend to offer coarse-grained controls that depend on a third-party assessment of the age-appropriateness of the app as expressed by an app content rating. Parental control systems typically allow a parent to specify that apps with certain ratings can be installed and others cannot. Inevitably, this leads to unintentional denial of certain apps. For example, a chess app may be rated as “mature” due to a misunderstanding on the part of the app developer. This is because current app ratings are based in part on answers that developers provide on a questionnaire. When acceptable content is inadvertently blocked, it is difficult to make an exception to the policy for the specific app. Parents must decide between loosening the policy restrictions and forgoing desired content. This kind of false positive can lead to parents disabling parental control systems. In addition to app install, mobile devices offer other features that pose risks. Communication channels, like SMS and phone calls, are useful for children but may expose children to cyber-bullying or predators. Parents may want to restrict the people that their child interacts with, or at least be aware of such interactions. However, many parental control systems either do not mediate these channels, or completely restrict them. Parental control systems should provide fine-grained controls that are simple to manage.
Parents are not always physically present with their children. By nature, mobile devices travel with their users. When parents are at work, and children are at school, at home, or playing with friends, mobile devices provide the connectivity needed to allow parents to actively protect their children. Yet, many parental control systems require the parent to have physical access to the child’s device to modify policy, or to access dedicated web applications to administer policy changes. This exasperates the problem of policy specification, because an incorrectly specified policy may result in an undesired denial of access to an operation that the parent would allow. If the parent and child are in separate locations, the child must wait until later to request a policy change from the parent. In the worst case, parental control systems are abandoned due to too many unintended denials of access.

Many mobile parental control systems can be bypassed. For a parental control system to be effective, it must mediate each access to a sensitive operation. To do this it must not be able to be bypassed by the child. Since current parental control applications are implemented as third-party apps, they are limited in their ability to control other apps. This is due to the design of mobile operating systems that employ protection mechanisms to isolate and sandbox apps from one another and the operating system. This is done primarily to contain malfunctioning apps or malware. In the case of parental control software, these mechanisms limit the effectiveness that can be achieved. The most effective existing parental control systems are based on the idea of device administration. Device administration provides system-level support to enforce certain policies and is an effective way to implement certain aspects of a parental control system, such as remote lock or protecting an application from being terminated by the user. However, device administration is targeted towards the employer-employee relationship, which is similar to but not the same as the parent-child relationship. In particular, device administration does not provide the level of sensitive-operation mediation that is desired for parental control systems.

Existing mobile parental control systems are not privacy-respecting. The parental control problem does not intrinsically require parents to disclose personal information about themselves or their children to a third party. Therefore, privacy-respecting systems are desirable. Parental control systems may store personal information on third-party systems in order to facilitate web interfaces for policy management, reporting, analytics, or billing. Web-based monitoring consoles are popular, but imply that sensitive information about the child and parent is being exposed to a third-party channel. While such information may be protected by the third party, the
user must relinquish control of their data and depend on the ability of the third party to secure it. As an external system becomes a repository for a large amount of sensitive data, that system becomes a more desirable target to would-be data thieves.

A mobile parental control system that is fine-grained, simple to administer, effective and privacy-respecting is desirable. Mobile devices, with their popularity, high connectivity, and processing capability provide an opportunity to provide such a parental control system by offering in-band policy management in a secure and privacy-respecting way.

4.2 Desired Properties

We now discuss the desired properties of a mobile parental control system that addresses the limitations just described. The system properties are shown in Table 4-1. The system properties are grouped into the three categories: usability, security, and privacy. The goal of a parental control system is to protect the child from exposure to content that the parent has deemed as potentially harmful. As such, the system is not used in isolation, but is integrated into the device that is being controlled. When the child is not at risk of exposure to harm, the parental control system should be out of the way. However, when a risk is detected, the system must do several things. First, it must block the risky operation. Second, it must inform the child that the operation was blocked. Third, the system should provide an intuitive way for children to request changes to the policy. Finally, the system must provide an intuitive way for the parent to respond to such requests.
Table 4-1. Desired System Properties

<table>
<thead>
<tr>
<th>Prop</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Usability</td>
<td>On-demand parental control</td>
</tr>
<tr>
<td>1.2</td>
<td>Usability</td>
<td>In-band policy management</td>
</tr>
<tr>
<td>1.3</td>
<td>Usability</td>
<td>Simple system setup</td>
</tr>
<tr>
<td>1.4</td>
<td>Usability</td>
<td>Simple policy management</td>
</tr>
<tr>
<td>2.1</td>
<td>Security</td>
<td>Availability</td>
</tr>
<tr>
<td>2.2</td>
<td>Security</td>
<td>Integrity</td>
</tr>
<tr>
<td>2.3</td>
<td>Security</td>
<td>Confidentiality</td>
</tr>
<tr>
<td>3.1</td>
<td>Privacy</td>
<td>Privacy-respecting design</td>
</tr>
</tbody>
</table>

_Usability_ refers to the ease of use of the system. The parent must be able to administer the system on-demand. Regardless of proximity to the child or whether or not the parent has physical access to the child device, the parental controls must be available. The child must be able to request policy modifications from the parent in-band. The system must be simple to setup. The system policy must be simple to administer.

_Security_ refers to the ability of the system to be available, that is, to deliver the desired functionality in the presence of inadvertent or deliberate threats. Also in view is the ability of the system to keep data confidential and to ensure that information is not leaked outside of the system to potential eavesdroppers. Finally, the system must ensure its integrity by correctly identifying system actors and properly attributing messages from them.

_Privacy_ refers to properties that respect the privacy of the users. Generally speaking, information should not be shared with third parties except as is necessary to provide the system functionality. The system should not store any readable user information on any system outside of the mobile devices used by the users. This property protects the users from potential adversaries gaining control of sensitive information used by the system, such as contacts, app lists, locations or the parent policy decisions.
Chapter 5

System Design

In this section we present the design of Protect My Child (PMC), a mobile parental control system. PMC delivers the desired system properties discussed in Chapter 4 and many other industry-standard features. The design presented in this chapter is a platform-independent design. The implementation-specific aspects of the design, which are based on the Android Open Source Project (AOSP), are discussed in Chapter 6. We begin by defining concepts and terms that will simplify the discussion. We then enumerate the system functional requirements. This is followed by a list of the specific challenges presented by the desired system properties from Chapter 4 and a discussion of the design principles that will be applied to meet those challenges. Finally, we present and discuss the system architecture.

5.1 Concepts

Before discussing the design of our system, we review several important concepts. The system actors consist of one or more parents and one or more children. A parent is an individual who has guardian authority and responsibility for a child. A child is an individual who is under the authority of a responsible guardian. Parent and Child actors may each have multiple mobile devices. A device is a smartphone or tablet running a mobile operating system such as Android, iOS, or Windows Mobile. A device designated for use by a parent is called a parent device and a device designated for use by a child is called a child device. Parent and child devices are grouped into logical groups where the parent devices interact with the child devices. This logical group is called a device family, and consists of one or more parent devices and one or more child devices. Figure 5-1 shows a diagram of a device family that consists of two parent devices and three child devices. Children perform operations on their devices. Some of these operations are considered sensitive operations. Sensitive operations are mediated and subject to policy defined by the device family parents. Policy dictates which operations are allowed and which are not.
5.2 Assumptions

It is assumed that a child device is a member of no more than one device family at a time. When there are multiple parents in a device family, it is assumed that all parents have equal authority. In all cases, the latest decision by a parent will be enforced on the child device.

5.3 Functional Requirements

The PMC system functional requirements are now given. They are given in the form of statements of what the system shall do or what system actors will be able to do. Thus, we have statements of the form, “The system shall …”, and of the form, “The system actor (user, parent, or child) shall be able to ….” The functional requirements are listed in Table 5-1.
### Table 5-1. Functional Requirements

<table>
<thead>
<tr>
<th>Requirement Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Setup</td>
<td>A parent shall be able to associate parent and child devices together into a device family.</td>
</tr>
<tr>
<td>Remote Device Release</td>
<td>A parent shall be able to remotely remove a child or parent device from a device family.</td>
</tr>
<tr>
<td>Local Mediation</td>
<td>The system shall mediate the sensitive operations specified in Table 5-2 locally, on the child device.</td>
</tr>
<tr>
<td>Remote policy management</td>
<td>A parent shall be able to remotely modify policy for each child device.</td>
</tr>
<tr>
<td>In-band policy management</td>
<td>A child shall be able to remotely request policy modifications from parents and the system shall alert all parents when a child requests a policy modification. Parents shall be able to approve or deny requests and the system shall immediately apply changes on to the child device policy.</td>
</tr>
<tr>
<td>Remote Location Tracking</td>
<td>A parent shall be able to remotely obtain the current location of child devices.</td>
</tr>
<tr>
<td>Emergency Notification</td>
<td>A child shall be able to send an emergency notification that contains the current time and the most recent available location of the child device to all parents.</td>
</tr>
<tr>
<td>Remote Lock</td>
<td>A parent shall be able to remotely lock and unlock a child device on demand. When remotely locking a child device, the parent shall be able to specify a lock delay and/or duration. The parent shall be able to specify a lock/unlock schedule for a child device.</td>
</tr>
<tr>
<td>Safe Search</td>
<td>A parent shall be able to remotely enforce web search filters on a child device.</td>
</tr>
<tr>
<td>Remote Contact White/Black Lists</td>
<td>A parent shall be able to remotely specify a white list and a black list of phone numbers for a child device.</td>
</tr>
<tr>
<td>Remote App Install and Uninstall</td>
<td>A parent shall be able to remotely install and uninstall an app on a child device.</td>
</tr>
<tr>
<td>Remote App Lock</td>
<td>A parent shall be able to remotely lock and unlock individual apps or a set of apps on the child device.</td>
</tr>
</tbody>
</table>

During the *system setup*, the parent will be able to create a new device family, or join an existing one. The child will only be able to join an existing family. The devices will exchange unique identifiers and public keys to enable secure communications over a public channel. The setup information will be exchanged using near-field-communication (NFC), or by scanning two-dimensional barcodes displayed by each device. This system setup exchanges security-sensitive information in a secure and privacy-respecting way. When a child device is not a member of a device family, the system will not enforce any policy or allow the device to be remotely administered. Essentially the parental control system will be switched off when a child device is not joined to a device family.
During *remote device release*, the parent must be able to remotely release a child device from the device family. This functionality will be available to any parent in the device family, but will not be available to any child. When a child device is released, it will revert to non-enforcing mode, allowing all operations. The system must ensure that a child that is joined to a device family cannot join another device family unless the child device is first released.

The child device shall provide *local mediation* of sensitive operations. Table 5-2 shows the operations that are considered sensitive. Each row in Table 5-2 lists an operation, its description and the granularity at which it is to be mediated.

**Table 5-2. Sensitive Operations**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS</td>
<td>Sending and receiving SMS messages</td>
<td>Phone number</td>
</tr>
<tr>
<td>PHO</td>
<td>Placing and receiving phone calls</td>
<td>Phone number</td>
</tr>
<tr>
<td>APPI</td>
<td>App Install/Uninstall</td>
<td>App</td>
</tr>
<tr>
<td>APPU</td>
<td>App Use</td>
<td>App</td>
</tr>
<tr>
<td>DEVU</td>
<td>Device Use</td>
<td>Device</td>
</tr>
</tbody>
</table>

SMS and phone operations are mediated at the phone-number level. This means that the child must have permission to send and receive messages and calls with each individual phone number. However, permission to message with parents is implicit. APPI operations include installing and uninstalling apps. App install and uninstall are mediated at the app level. This means permission is required to install or uninstall each app. APPU operations include launching apps. Like APPI operations, APPU operations are mediated at the app level. App use is initially allowed for all apps on the child device, but can be restricted by the parent. DEVU operations include use of the device. A parent denies device usage by remotely locking the device. A locked device cannot be used by the child except to contact a parent or an emergency number. Device use is mediated at the device level.

*Remote policy management* allows the parents be able to remotely manage the child’s access control policy from their parent device. Local policy management on the child device is excluded. Including local policy management features on the child device adds an unnecessary
attack surface to the system. If such features exist on the child device, the child may be able to gain access to them and bypass the system.

In-band policy management allows children to request authorization to access a mediated operation when it is denied. On the child device, the system will enforce the policy set by the parents. When the child attempts to perform a blocked operation, the system will prevent the operation and notify the child that the operation was blocked. The child will be able to request permission to access the blocked operation from the notification. If the child requests permission to perform a denied operation, the parent will be notified immediately. The system will allow the parent to view the details of the operation requested and approve or reject it. Once the parent has approved or rejected the request, the child is immediately notified of the result. If the request was approved, the access control policy on the child device will be permitted to perform that operation from that point on, unless the parent later revokes the permission. The child will not be blocked on subsequent attempts to access the same operation that was approved by a parent. If multiple parents respond with different decisions, the latest decision will take effect on the child device.

The parents are able to perform remote location tracking of a child device from a parent device. When the parent requests the child device location, the child device will attempt to obtain its current location. If it is unable to obtain its location, it will use the last location that the device was able to obtain. If no location can be obtained, the child device will respond with an error. When a location is reported to the parent device, the location will include the time at which it was measured and an error radius.

The child will be able to send an emergency notification to all parent devices using an interface provided on the child device. The emergency notification will include the time and most recent available location of the device. If no current or last location is available, the system will send an emergency notification to the parent devices that does not include the location.

The parent will be able to perform a remote lock of the child device from the parent device. Locking the child device will render it unusable by the child until the device is unlocked by the parent. A remotely locked child device will still allow the child to call the parent, emergency phone numbers (like 911) and send emergency notifications to the parent. When the parent remotely locks the child device, they will be able to specify a lock-duration. Once the duration
has expired, the child device will automatically unlock. The child device will display the unlock
time and the amount of time until the device will unlock. When the parent remotely locks the
child device, they will be able to specify a lock delay. The child device will wait for the duration
of the specified delay before locking. The child device will periodically notify the child that it
will be locking and indicate how much time is remaining before the device locks.

The parent will be able to remotely specify a lock schedule for the child device using an interface
on the parent device. The schedule will allow the parent to specify days of the week and times of
the day when the device will be locked and unlocked. The lock schedule will be sent to the child
device and the child device will enforce the schedule. The child device will periodically notify
the child when the device will be locking soon. When the device is locked, it will display the
amount of time left until it will unlock.

Some internet search providers, including Google and Bing, provide filtered search functionality,
called SafeSearch, which filters out results with explicit material like pornography. The parent
will be able to remotely turn on SafeSearch for these search providers from an interface on the
parent device.

The parent will be able to specify a contact white list and black list. The contact white list is a list
of phone numbers with which the child may freely communicate via SMS messages and phone
calls. The contact black list is a list of contacts with which the child may not communicate.
When the child attempts to exchange SMS messages with or call contacts on the black list, they
will be blocked. The system will ensure that the same phone number does not appear on both the
white list and the black list at the same time.

Remote app install and uninstall will allow the parent to remotely install an app on the child
device. The parent will be able to select apps that will be remotely installed on the child device
without any action required of the child. The parent will be able to remotely view the apps that
are installed on the child device and remotely uninstall apps.

The remote app lock feature will allow a parent to view the apps that are installed on the child
device and remotely lock an individual app or a set of apps. A locked app will not be able to be
run on the child device. The list of installed apps will be annotated with the app lock state (locked or unlocked).

5.4 System Architecture

The PMC system architecture is shown in Figure 5-2. There are three kinds of devices in the system. The parent and child devices are mobile devices. The Message Server and CMS are hosted on internet accessible servers. The CMS is not part of PMC, but is a service provided by a CMS provider such as Google and Apple.

![Figure 5-2. System Architecture](image)

The Parent App is a typical user app that can be installed on a commodity mobile device running a commodity operating system. It provides user interfaces to the parents for managing the device.
family and administering the child devices. The Parent App allows the user to create a device family, join and release devices with the device family and reset the entire system. The Parent App also notifies the parent of events that occur on the child device including permission requests, and emergency notifications. The Parent App communicates with other devices in the device family via the Message Server. All messages sent to the message server are protected using digital envelopes and digital signatures as described below in 5.8 Security and Privacy.

The Message Server is a web application that runs in the cloud to facilitate messaging between the devices in a device family. The Message Server implements a REST interface for registering a device, sending a message and retrieving a message. Messages are posted to the Message Server for a particular recipient and are stored in a FIFO queue until that recipient retrieves the message. The Message Server REST API is shown in Table 5-3. Access to the Message Server is restricted to clients that possess a certificate trusted by the server. The server also presents a certificate to the client.

Table 5-3. Message Server REST API

<table>
<thead>
<tr>
<th>Verb</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT</td>
<td><code>/msg/{recipient-id}</code></td>
<td>Enqueues the request payload as a message to <code>{recipient-id}</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/msg/{recipient-id}</code></td>
<td>Dequeues a message for <code>{recipient-id}</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/msg/register</code></td>
<td>Registers a recipient ID with a cloud messaging ID, both contained in the message payload</td>
</tr>
</tbody>
</table>

When a message is posted to the message server, it must know which device should receive the message. For this reason, the message sent to the message server includes a unique ID that identifies the intended recipient. This ID is the PMC ID generated by PMC during system setup.

The message server enqueues the message in a queue maintained for the intended recipient. Once the message has been enqueued, the message server sends a notification to the recipient via the CMS. The purpose of this message is merely to inform the recipient that a message has been enqueued for it at the message server. In order to use a CMS, each device must register with the CMS. At registration time, the device receives a CMS ID. The messaging server must know the recipient CMS ID in order to request that a message be pushed to the recipient device. Therefore the messaging service stores a map from PMC ID to CMS ID.
The *Child App* is a privileged user app that runs on child devices. The Child App provides user interfaces to notify the child when an operation is blocked, allow the child to request policy changes, and notify the child of the parent’s decisions. The Child App also provides a user interface for the child to send emergency notifications to the parents.

The Child App receives messages from the CMS and retrieves messages from the Message Server. It verifies capability signatures for policy changes from the parents and applies changes to the policy store. It also provides an authentication module service to the reference monitor. Because the Child App is a privileged app, it cannot be terminated or uninstalled by the user. Therefore the authentication module service and policy store are always available to the PMC reference monitor.

In order to ensure that all sensitive operations are completely mediated, the child device operating system is hooked. These *system hooks* are inserted into the framework and call on the reference monitor to obtain authorization decisions. Based on the decision of the *reference monitor*, the hooked operation either continues or is disrupted. The reference monitor runs within the device operating system and delegates authorization decisions to the authorization module running within the Child App. The interaction between the operating system, reference monitor and Child App is shown in Figure 5-3.

![Figure 5-3. Mediation of Operations](image-url)
5.5 Security Model

The TCB was shown in Figure 5-2. The Parent App and therefore the operating system on the parent device are trusted. The cloud services are not trusted and all message contents are encrypted before being handled by the message server. On the child device, the Child App and the operating system are trusted. This includes the reference monitor and framework hooks to be implemented in the device operating system.

The threats we are most concerned with are those that would disable or degrade the parental controls of the system and those that would disclose private data to a third party. These attacks could originate with the child actor or a third-party attacker.

The child actor has physical access to device, but does not have root access. The child actor could attempt to disable PMC by force closing or uninstalling the PMC app. The child may also attempt to bypass PMC by rebooting the device into “safe-mode”, or performing a factory reset of the device. The child could prevent connectivity with the parent device by disabling network communications on the device.

A third-party attacker does not have physical access to the device, but could gain access to messages between parent and child devices. A third-party attacker could attempt to impersonate a parent to a child or replay messages that have already been sent to the child device in an attempt to track the child device location or gain SMS or phone call permissions.

5.6 Design Principles

Given the system functional requirements from Table 5-1, each of the system properties listed in Table 4-1 present certain challenges to the system design. In Table 5-4, the design properties are repeated and the challenges that they present are enumerated. Each design challenge is then mapped to design principles that will be used to address those challenges.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property 1.1:</strong></td>
<td>On-demand parental control</td>
</tr>
<tr>
<td><strong>Challenge 1.1.1:</strong></td>
<td>Allow interaction when parent and child are on different networks, in different locations</td>
</tr>
<tr>
<td><strong>Design Principle 1.1.1:</strong></td>
<td>Use a CMS</td>
</tr>
<tr>
<td><strong>Property 1.2:</strong></td>
<td>In-band policy management</td>
</tr>
<tr>
<td><strong>Challenge 1.2.1:</strong></td>
<td>Allow child to request immediate policy changes from parent</td>
</tr>
<tr>
<td><strong>Design Principle 1.2.1.1:</strong></td>
<td>Request/response-based access control</td>
</tr>
<tr>
<td><strong>Property 1.3:</strong></td>
<td>Simple system setup</td>
</tr>
<tr>
<td><strong>Challenge 1.3.1:</strong></td>
<td>Exchange identity information when associating devices</td>
</tr>
<tr>
<td><strong>Design Principle 1.3.1.1:</strong></td>
<td>Match mental model of associating devices</td>
</tr>
<tr>
<td><strong>Property 1.4:</strong></td>
<td>Simple policy management</td>
</tr>
<tr>
<td><strong>Challenge 1.4.1:</strong></td>
<td>Express fine-grained control using simple policy specification language</td>
</tr>
<tr>
<td><strong>Design Principle 1.4.1.1:</strong></td>
<td>Economy of mechanism (keep policy design as simple as possible)</td>
</tr>
<tr>
<td><strong>Property 2.1:</strong></td>
<td>Availability</td>
</tr>
<tr>
<td><strong>Challenge 2.1.1:</strong></td>
<td>Prevent child from disabling system</td>
</tr>
<tr>
<td><strong>Design Principle 2.1.1.1:</strong></td>
<td>Isolated Reference Monitor</td>
</tr>
<tr>
<td><strong>Challenge 2.1.2:</strong></td>
<td>Prevent child from bypassing system</td>
</tr>
<tr>
<td><strong>Design Principle 2.1.2.1:</strong></td>
<td>Complete mediation</td>
</tr>
<tr>
<td><strong>Challenge 2.1.3:</strong></td>
<td>Prevent child from modifying policy</td>
</tr>
<tr>
<td><strong>Design Principle 2.1.3.1.1:</strong></td>
<td>Least privilege – Child has no need to modify policy, so will be unable to do so</td>
</tr>
<tr>
<td><strong>Property 2.2:</strong></td>
<td>Integrity</td>
</tr>
<tr>
<td><strong>Challenge 2.2.1:</strong></td>
<td>Prevent impersonation of parents and children</td>
</tr>
<tr>
<td><strong>Design Principle 2.2.1.1:</strong></td>
<td>Verify policy changes are authentic</td>
</tr>
<tr>
<td><strong>Design Principle 2.2.1.2:</strong></td>
<td>Verify message are authentic</td>
</tr>
<tr>
<td><strong>Property 2.3:</strong></td>
<td>Confidentiality</td>
</tr>
<tr>
<td><strong>Challenge 2.3.1:</strong></td>
<td>Prevent eavesdropping of messages between parent and child</td>
</tr>
<tr>
<td><strong>Design Principle 2.3.1.1:</strong></td>
<td>Use message encryption</td>
</tr>
<tr>
<td><strong>Design Principle 2.3.1.2:</strong></td>
<td>Exchange identity information over a trusted channel</td>
</tr>
<tr>
<td><strong>Property 3.1:</strong></td>
<td>Privacy-respecting design</td>
</tr>
<tr>
<td><strong>Challenge 3.1.1:</strong></td>
<td>Prevent third-party access to user data</td>
</tr>
<tr>
<td><strong>Design Principle 3.1.1.1:</strong></td>
<td>Minimize trusted computing base (TCB), restrict TCB to system actors.</td>
</tr>
</tbody>
</table>
The design challenges and principles are discussed in the following sections on usability, security and privacy. These are the system design property categories identified in Chapter 4.

5.7 Usability

Usability refers to the ease of use of a system. The PMC system design must provide the four usability properties from Chapter 4. These are repeated in Table 5-5.

Table 5-5. System Usability Design Properties

<table>
<thead>
<tr>
<th>Prop</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Usability</td>
<td>On-demand parental control</td>
</tr>
<tr>
<td>1.2</td>
<td>Usability</td>
<td>In-band policy management</td>
</tr>
<tr>
<td>1.3</td>
<td>Usability</td>
<td>Simple system setup</td>
</tr>
<tr>
<td>1.4</td>
<td>Usability</td>
<td>Simple policy management</td>
</tr>
</tbody>
</table>

These design properties will be provided by adhering to the four design principles enumerated above and repeated in Table 5-6.

Table 5-6. System Usability Design Principles

<table>
<thead>
<tr>
<th>Design Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.1</td>
<td>Use a CMS</td>
</tr>
<tr>
<td>1.2.1.1</td>
<td>Request/response-based access control</td>
</tr>
<tr>
<td>1.3.1.1</td>
<td>Psychological acceptability</td>
</tr>
<tr>
<td>1.4.1.1</td>
<td>Economy of mechanism</td>
</tr>
</tbody>
</table>

The first usability property is that the system will provide on-demand parental control. This means that parents will be able to control the child device remotely, from their own mobile devices. Such control requires connectivity. Mobile devices offer Wi-Fi and cellular data network connectivity to the internet. The system will rely on internet connectivity to communicate between devices. Although mobile devices have internet connectivity, it is impractical to communicate directly from device to device. This is because mobile devices change networks often, resulting in different IP addresses. They are often behind network address translation (NAT) devices which complicate establishment of incoming connections. For these reasons, it is difficult to address the device. Instead, a message server is used to provide a
static service that all devices with internet connectivity can reach. Due to power constraints, it is not desirable that the mobile devices should poll the message server for new messages. To address this concern, a CMS is employed. A CMS is able to efficiently push notifications to mobile devices.

The architecture to support on-demand parental control is highlighted in Figure 5-4. When a message is to be sent from the parent device to a child device, the parent will send the message to the message server. The message server will store the message for the recipient, the child device in this case, and send a request to a CMS to notify the child that a message is available. The CMS will push a notification to the child device. Upon receiving this notification, the child device will make a request to the message server to obtain the message. When a message must be sent from the child device to a parent device, the protocol is reversed. This approach provides the connectivity required to provide on-demand parental control in a power-efficient way.

![Figure 5-4. Device Communication](image)

The second usability property is in-band policy management. This means that the child can request changes to the parental control policy from within the system and the parent can respond immediately to the requests. The interaction diagram in Figure 5-5 shows the process of performing a sensitive operation, in this case, installing an app on the child device.
The child attempts to install an app using the functionality provided by the mobile device, for example from the Google Play store on an Android device or the App Store on an iOS device. The application installation is performed by the mobile device operating system framework, which is hooked to mediate this operation. The hooks call into the PMC system, where the child’s right to install the app is checked by examining the current policy. If the child has not been granted the right to install the particular app, a denial is returned by from PMC to the framework hooks, terminating the installation process. PMC also generates a notification to the child informing them that the installation was blocked and providing an interface with which the child can request permission from the parent to install the app. If the child decides to ask for permission, the system immediately sends a notification to the parent device. On the parent device, the permission request is displayed as a notification, similar to when an SMS or email is received. When the parent opens the notification, the details of the request are shown and the parent can decide whether or not to allow the install to proceed. If the parent approves the install request, PMC sends a capability back to the child device that allows the Child App to update the PMC policy to allow the app to be installed. The child is immediately notified of the parental
decision and the app is then automatically installed. An analogous sequence occurs for SMS messaging and phone calls.

The third usability property is *simple system setup*. Before the parent can administer the child device, the devices need to be associated together into a device family. Only a parent can create a device family. This is accomplished using an interface on the parent device. Once a device family has been created, a child (or another parent) can join the device family. This operation associates the devices together. Each device is identified by a universally unique identifier (UUID) that is generated by PMC. We refer to this as a PMCID. When a device joins a device family, it provides its PMCID to the parent, and the parent device provides its PMCID to the child. In addition, the information shown in Figure 5-6 is exchanged.

![Figure 5-6. Adding a Device to a Device Family](image)

The Join Request and Join Response messages contain sensitive information and must be exchanged over a secure channel. If an attacker was able to intercept the Join Request or the Join Response, or both, they would be able to eavesdrop on messages or perform a man-in-the-middle attack. To exchange these messages, PMC will use temporary, private, close-proximity channels based on NFC or 2-D barcodes scanned by mobile device cameras.
The final usability system property is *simple policy management*. Fine-grained policy management is a major feature of PMC. However, complicated policy management can lead to system abandonment. PMC offers fine-grained policy control by requiring permission for each operation that was defined in Table 5-2. Initially, the child has no permission to perform sensitive operations, with the exception of device and app usage. SMS, phone call and app install/uninstall is initially denied. As the child makes new contacts, or finds new apps to install, the child makes a one-time permission request for access to each operation. The in-band policy management facilitates this approach and keeps it simple, mimicking the natural request-response relationship that parents and children already share. As permission is granted for each app or contact, the access control policy is updated on the child device, so that permitted operations are not denied in the future, until permission is revoked by the parent.

In addition to fine-grained access control, the system offers coarse-grained controls for apps and contacts. For apps, three levels of adaptive controls are offered: allow all, allow age-appropriate, and request each. Allow-all mode allows the child to install and uninstall any user app without the need for explicit permission. The PMC Child App is not considered a user app and cannot be uninstalled while the device is in a device family. When allow-all mode is on, the app install/uninstall permissions are not checked for app install and uninstall operations.

In order to support age-appropriate mode, the parent must enter a birth date for the child. App marketplaces typically contain a rating for each app that can be mapped to an appropriate age range. For example, Android apps in the Google Play Store have ratings of Everyone (E), Everyone 10 and Up (E10), Teen (T), Mature (M) and Adult Only (AO). These ratings correspond to ages of 0+, 10+, 13+, 17+, and 18+, respectively. In age-appropriate mode, the child may install any app with a rating that is appropriate for the child’s age, based on the provided birth date and the app rating, without the need for explicit permission. However, if the child attempts to install an app that is not age-appropriate, the PMC app permissions are checked and the system blocks apps that are not explicitly permitted. In such cases, the child can request permission to install apps using the in-band policy management. The final mode, request-each, requires that the child have explicit permission to install or uninstall each app. The app mediation modes are summarized in Table 5-7.
Table 5.7. App Mediation Modes

<table>
<thead>
<tr>
<th>App Mediation Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request-each</td>
<td>The child must have explicit permission to install or uninstall each user app.</td>
</tr>
<tr>
<td>Age-appropriate</td>
<td>The child has implicit permission to install and uninstall age-appropriate user apps. The child must have explicit permission to install or uninstall other user apps.</td>
</tr>
<tr>
<td>Allow-all</td>
<td>The child has implicit permission to install or uninstall any user app.</td>
</tr>
</tbody>
</table>

For SMS and phone calls, the system offers two levels of coarse-grained controls: allow-all and request-each. These contact mediation modes are analogous to the app mediation modes described above. In allow-all mode, the child is implicitly permitted to call or SMS with any number. In request-each mode, the child must have explicit permission to interact with each contact. The contact mediation modes are summarized in Table 5.8.

Table 5.8. Contact Mediation Modes

<table>
<thead>
<tr>
<th>Contact Mediation Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request-each</td>
<td>The child must have explicit permission to call or SMS with each contact.</td>
</tr>
<tr>
<td>Allow-all</td>
<td>The child has implicit permission to call or SMS with each contact.</td>
</tr>
</tbody>
</table>

5.8 Security and Privacy

Because security and privacy are closely related, the security and privacy properties are addressed together. The design of the PMC system must provide the three security properties and the one privacy property from Chapter 4. Those properties are repeated in Table 5.9. The system will employ the design principles shown in Table 5.10 to achieve these properties.
Table 5-9. Security and Privacy Properties

<table>
<thead>
<tr>
<th>Prop</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Security</td>
<td>Availability</td>
</tr>
<tr>
<td>2.2</td>
<td>Security</td>
<td>Integrity</td>
</tr>
<tr>
<td>2.3</td>
<td>Security</td>
<td>Confidentiality</td>
</tr>
<tr>
<td>3.1</td>
<td>Privacy</td>
<td>Privacy-respecting design</td>
</tr>
</tbody>
</table>

Table 5-10. Security and Privacy Design Principles

<table>
<thead>
<tr>
<th>Design Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1.1</td>
<td>Isolation of reference monitor</td>
</tr>
<tr>
<td>2.1.2.1</td>
<td>Complete mediation</td>
</tr>
<tr>
<td>2.1.3.1</td>
<td>Least privilege</td>
</tr>
<tr>
<td>2.2.1.1</td>
<td>Authentic policy changes</td>
</tr>
<tr>
<td>2.2.1.2</td>
<td>Authentic messages</td>
</tr>
<tr>
<td>2.3.1.1</td>
<td>Encrypted messages</td>
</tr>
<tr>
<td>2.3.1.2</td>
<td>Exchange identity information over a secure channel</td>
</tr>
<tr>
<td>3.1.1.1</td>
<td>Minimize the system TCB</td>
</tr>
</tbody>
</table>

The system architecture diagram in Figure 5-2 showed the TCB components of the PMC system as shaded boxes. The operating systems of the parent and child devices are trusted, as well as the Parent App and Child App. The operating system must provide isolation guarantees between apps. Both Android and iOS provide application sandboxing to isolate app data from other apps. On the child device, the system hooks and reference monitor are implemented in the system framework. This is necessary to ensure that the reference monitor cannot be bypassed and can provide complete mediation of sensitive operations. This guarantees that whenever the system is running, the reference monitor is running as well.

As a system component, the reference monitor is isolated, that is, it is not subject to modification by software running outside of the TCB. PMC implements the reference monitor as a system service which delegates access control decisions to an authorization module inside of the trusted PMC Child App. The PMC Child App stores the access control policy in a local database. This policy database is controlled by the PMC Child App and is also part of the TCB. Updates to the policy database require a capability token that has been digitally signed by a parent member of the device family. Consistent with the design principle of least privilege, the Child App does not have the ability to generate these tokens on its own because it does not have access to the parent private keys.
The access control policy is used by the reference monitor to answer authorization queries. An advantage of a local policy store is that the child device can be used with the most recent policy settings even when the device cannot communicate with the internet. Using a local policy store provides fail-safe defaults without sacrificing availability.

Policy changes must be authenticated. As mentioned above, when a parent issues a policy change, a special capability token will be generated. The capability token consists of the fields shown in Table 5-11. The token includes an increasing sequence number that prevents replay attacks, and includes a digital signature that is created by the Parent App and verified on the Child App. The digital signature guarantees the authenticity of the capability.

### Table 5-11. Policy Change Capability Token Fields

<table>
<thead>
<tr>
<th>Capability Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td>A sequence number that increases for each capability issued</td>
</tr>
<tr>
<td>Capability ID</td>
<td>A UUID identifying the capability</td>
</tr>
<tr>
<td>Parent PMCID</td>
<td>A UUID assigned to the parent device by PMC</td>
</tr>
<tr>
<td>Right</td>
<td>The right that this capability grants (See Table 5-12)</td>
</tr>
<tr>
<td>Child Device PMCID</td>
<td>A UUID assigned to the child device by PMC</td>
</tr>
<tr>
<td>Object</td>
<td>The system object the right pertains to (See Table 5-13)</td>
</tr>
<tr>
<td>Parameters</td>
<td>Parameters based on the right and object</td>
</tr>
<tr>
<td>Digital Signature</td>
<td>All previous fields are delimited, concatenated and digitally signed by the PMC Parent Device that is issuing the capability</td>
</tr>
</tbody>
</table>
Table 5-12. Rights

<table>
<thead>
<tr>
<th>Right</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS_SENDRECV</td>
<td>Send and receive with an SMS contact</td>
</tr>
<tr>
<td>SMS_SENDRECV_DENY</td>
<td>Deny send and receive with an SMS contact</td>
</tr>
<tr>
<td>APP_INSTRMV</td>
<td>Install and uninstall an app</td>
</tr>
<tr>
<td>APP_INSTRMV_DENY</td>
<td>Deny install and uninstall an app</td>
</tr>
<tr>
<td>APP_ACCESS</td>
<td>Allow to run an app</td>
</tr>
<tr>
<td>APP_ACCESS_DENY</td>
<td>Deny running an app</td>
</tr>
<tr>
<td>BROWSER_FORCE_SAFE_SEARCH</td>
<td>Turn on browser SafeSearch</td>
</tr>
<tr>
<td>BROWSER_NO_FORCE_SAFE_SEARCH</td>
<td>Turn off browser SafeSearch</td>
</tr>
<tr>
<td>PHONE_LOCK</td>
<td>Lock the phone screen</td>
</tr>
<tr>
<td>PHONE_UNLOCK</td>
<td>Unlock the phone screen</td>
</tr>
<tr>
<td>PHONE_LOCK_SCHED_ENTRY</td>
<td>Add a lock schedule entry</td>
</tr>
<tr>
<td>PHONE_REMOVE_LOCK_SCHED_ENTRY</td>
<td>Remove a lock schedule entry</td>
</tr>
<tr>
<td>FAM_ADD</td>
<td>Add a family member to device family</td>
</tr>
<tr>
<td>FAM_RMV</td>
<td>Remove a family member from device family</td>
</tr>
</tbody>
</table>

Technically, these are rights to modify policy on the child device.

Table 5-13. Objects

<table>
<thead>
<tr>
<th>System Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS</td>
<td>SMS Messaging permission changes</td>
</tr>
<tr>
<td>CALL</td>
<td>Phone Call permission changes</td>
</tr>
<tr>
<td>APP</td>
<td>App installing, removing and usage permission changes</td>
</tr>
<tr>
<td>BROWSER</td>
<td>Web browsing permission changes</td>
</tr>
<tr>
<td>PHONE</td>
<td>Device usage permission changes</td>
</tr>
<tr>
<td>FAMILY</td>
<td>Changes to device family members</td>
</tr>
</tbody>
</table>

In order to satisfy the system confidentiality and privacy properties, all inter-device message payloads are encrypted using digital envelopes. To satisfy the system integrity property, the messages are also digitally signed. The required public keys are exchanged over a secure NFC or 2-D barcode channel during system setup.
Chapter 6

System Implementation

In the previous chapter, we described the system design without specific implementation details. In this chapter, the implementation of a PMC prototype for Android is described. The design described in Chapter 5 was implemented with some exceptions. As the design is described at the implementation level, these exceptions are pointed out.

6.1 Technologies

The PMC prototype was implemented for the Android operating system. This was primarily due to the open nature of the Android Open Source Project (AOSP) and the availability of inexpensive developer devices. The system was initially developed for Android 4.3 (Jelly Bean) and later ported to Android 5.0.1 release 1 (Lollipop). With this version of AOSP, the PMC AOSP build was run on Nexus 4, Nexus 5, Nexus 6, and Nexus 10 devices. Google Cloud Messaging (GCM) is used as the CMS. GCM is the dominant CMS for Android devices in the United States.

6.2 System Hooks

The system hooks are modifications to the Android operating system deployed to the child device. These hooks are light-weight and of the form shown in Figure 6-1. Rather than implementing authorization code in the framework, the hooks simply call on a reference monitor that is running as a system service. The reason for this design was to minimize the changes to the operating system. The data sent to the reference monitor depends on the operation that is being mediated. For SMS messages, the phone number of the contact and the text of the message are sent. For Phone calls, the phone number is sent. For app install/uninstall and app usage the app package name is sent.
if( refMon.isXYZAllowed(data) ) {
    // Perform the operation
} else {
    // Skip or disrupt the operation
    // Throw an exception or return an error
}

Figure 6-1. Structure of a System Hook

All of the hooks in the AOSP codebase are located in the system framework. The sending of SMS messages is hooked in the IccSmsInterfaceManager where access control permissions are checked. There are two functions in IccSmsInterfaceManager that send an SMS message depending on whether the message is multi-part or not: sendText and sendMultipartText. In both cases, if PMC denies the SMS send operation, a SecurityException is thrown. The hook code is shown in Figure 6-2.

... boolean allowed=mPpcsAccessManager.isSmsSendAllowed(destAddr,text);
if(!allowed) {
    throw new SecurityException("SMS send to "+destAddr+" denied by PCPCS");
}
...

Figure 6-2. SMS Send Hook in IccSmsInterfaceManager

Receiving SMS messages is hooked in InboundSmsHandler.java in the dispatchNormalMessage method. This is where incoming SMS text messages are dispatched from radio-specific code. The SMS receive hook code is shown in Figure 6-3. When an inbound text is not allowed, the message is marked as handled and dropped. This prevents the SMS message from being stored and displayed by the device.
boolean allowed=mPcpcsAccessManager.isSmsReceiveAllowed(address,messageBody);
if(!allowed) {
    // Mark the message handled, otherwise the fw will keep resending it.
    return Intents.RESULT_SMS_HANDLED;
}

Figure 6-3. SMS Receive Hook in InboundSmsHandler

Placing outgoing calls is hooked in PhoneProxy.java in the dial method. All outgoing calls go through this method where they are dispatched to the currently active phone. The hook code is shown in Figure 6-4. In the outgoing call hook, it is important to consider that the phone number could be an emergency number. PMC will never block a call to an emergency number. The AOSP class, PhoneNumberUtils, has an existing method called, isEmergencyNumber, that takes a dial string as an argument and indicates if the given dial string represents an emergency number. This method is called before calling on PMC to mediate the outgoing call. Only if isEmergencyNumber returns false, is PMC given the opportunity to mediate the call. When the number being dialed is not an emergency number, PMC is invoked to render an authorization decision. When the call is not allowed, a CallStateException is thrown indicating that the call is not allowed. This prevents the outgoing call.

if(!PhoneNumberUtils.isEmergencyNumber(dialString)) {
    boolean allowed = mPcpcsAccessManager.isCallSendAllowed(dialString);
    if(!allowed) {
        throw new CallStateException("Call not allowed");
    }
}

Figure 6-4. Dial Call Hook in PhoneProxy

Receiving incoming calls is hooked in PhoneBase.java, the base class for all phones. The hook is installed in the handleMessage method. The hook code is shown in Figure 6-5. The original code is shown in bold. When an incoming call is received, the number is mediated by PMC. If the receive call operation is denied, the call is automatically rejected. The child will hear the initial ring of the call and will be notified that the call was rejected. At that point the child can
use the in-band policy management to request permission to receive calls from that number. Private calls do not provide a phone number. When a call is received from a private number, the string, “Unknown” is used as the phone number. The child may request, and the parent may approve calls from private numbers. Note that this permission would apply to all private numbers.

```java
boolean allowed = true;
String number = getRingingNumber();
if(number!=null) {
    allowed = mPcpcsAccessManager.isCallReceiveAllowed(number);
}
if(allowed) {
    if (getState() == PhoneConstants.State.RINGING) {
        sendIncomingCallRingNotification(msg.arg1);
    }
} else { // if number is not allowed, automatically reject the call.
    try {
        CallManager.getInstance().
            rejectCall(CallManager.getInstance().
                        getFirstActiveRingingCall());
    } catch (CallStateException ex) {
        Rlog.d(LOG_TAG, "Exception rejecting call: " + ex);
    }
}
```

Figure 6-5. Receive Call Hook in PhoneBase

App installs and uninstalls are hooked in the Android Package Manager, PackageManagerService.java. App installs are hooked in two places within the package manager. The first hook is in the installNewPackageLI method, shown in Figure 6-6. When PMC denies an app install operation the result is set to an error value of INSTALL_FAILED_USER_RESTRICTED. The hook also saves off the APK that was being installed and sends the file path to PMC. This information is used by PMC to automatically install the app if the parent later approves it.
boolean allowed = mPcpcsAccessManager.isAppInstallAllowed(pkgName);
if (!allowed) {
    File dir = new File(pkg.codePath);
    File newName = new File(pkg.codePath + ".bak");
    if (dir.isDirectory()) {
        dir.renameTo(newName);
    }
    mPcpcsAccessManager.
        addPackagePath(pkg.codePath + ".bak/base.apk", pkgName);
    res.setError(INSTALL_FAILED_USER_RESTRICTED, "Package couldn't be installed because of the denial from PCPCS!");
    return;
}

Figure 6-6. First App Install Hook in PackageManagerService

The second app install hook is in the installExistingPackageAsUser method. The hook code is shown in Figure 6-7. When PMC denies an app install operation, the hook returns INSTALL_FAILED_USER_RESTRICTED. This prevents the install of the app.

... boolean allowed = mPcpcsAccessManager.isAppInstallAllowed(packageName);
if (!allowed) {
    return PackageManager.INSTALL_FAILED_USER_RESTRICTED;
}
...

Figure 6-7. Second App Install Hook in PackageManagerService

App uninstalls are also hooked in the package manager. The app uninstall is hooked in the deletePackage method. The hook code is shown in Figure 6-8. When PMC denies the uninstall operation, the hook sends the DELETE_FAILED_USER_RESTRICTED error to the uninstall observer and then exits.
boolean allowed = mPcpcsAccessManager.isAppUninstallAllowed(packageName);
if (!allowed) {
    try {
        observer.onPackageDeleted(packageName,
        PackageManager.DELETE_FAILED_USER_RESTRICTED, null);
    } catch (RemoteException re) {}
    return;
}

Figure 6-8. App Uninstall Hook in PackageManagerService

App launches are hooked in ActivityManagerService.java in the startProcessLocked method. The
ActivityManagerService is responsible for launching activities in Android. The hook code is
shown in Figure 6-9. If PMC denies the app launch operation, the hook does not launch the app
and returns null.

...  
if (!mPcpcsAccessManager.isAppAccessible(info.packageName)) {
    return null;
}
...

Figure 6-9. Launch App Hook in ActivityManagerService

6.3 Reference Monitor

The reference monitor is deployed as part of the operating system of the child device. It is an
Android system service defined in frameworks/base/core/java/android/os and created in
ContextImpl.java. Since it is a system service, the reference monitor is always available to the
System Hooks. The reference monitor policy store and authorization module exist in a service in
the Child App and are used by the reference monitor system service. The reference monitor
tolerates the absence of the Child App. In such cases, it allows all operations. Also, if the Child App is not part of a device family, the reference monitor allows all operations.

6.4 Child App

The Child App is deployed on each child device. The child device runs a modified version of the Android operating system built from AOSP. The Child App was implemented as an Android system app, developed in Android Studio. It is signed with the AOSP system key which enables it to use the FORCE_STOP_PACKAGES and GET_TASKS permissions. The Child App registers itself as a device administrator which protects it from being force-closed and allows it to lock the device screen. The Child App is targeted at Android 4.1 and later.

6.5 Parent App

An instance of the Parent App is deployed on each parent device. The Parent App is implemented as a standard, user-level app, developed using Android Studio. The Parent App targets Android 2.1 and later. The Parent App does not require any framework changes and can be installed on any Android 2.1 or later device.

6.6 Message Server

The Message Server facilitates communications between the devices in the PMC device family. To support the communications required by PMC, a simple design was adopted. The Message Server simply acts as a set of message queues that stores messages for designated recipients. The Message Server integrates with Google Cloud Messaging (GCM) to alert recipients that a message is available to be picked up. The only clear text information stored on the Message Server is the PMCID of each device and the GCM token associated with it. In theory, the Message Server could also store information concerning which PMCID's belong to devices that communicate with each other. Since a sending PMC app specifies the recipient by PMCID, each
time a message is sent an association could be made. The best possible result of this kind of
information gathering would be to identify the PMCIDs of the devices in a device family. The
PMC apps never send clear text messages, so no private information about the children or parents
can be collected.

The Message Server is implemented as a web application that provides the REST web service
described in Chapter 5. The application uses the Jedis library (https://github.com/xetorthio/jedis)
to communicate with a local Redis in-memory database. The Message Server uses GCM as the
CMS. The Message Server web application is hosted by a Tomcat application server. The
Tomcat server is configured to require mutually authenticated TLS connections. The Tomcat
Server is hosted by a server on Amazon Web Services.

6.7 Confidentiality and Integrity

The system confidentiality properties are satisfied using digital envelopes. The digital envelope
implementation uses 128 bit AES for the symmetric key and 1024 bit RSA for the asymmetric
keys. Digital signatures are implemented with 1024 bit RSA and the MD5 hash algorithm.

6.8 Functional Requirements

In this section implementation specific aspects of the functional requirements are highlighted.

System setup is implemented using one of two methods offered by PMC. The first method uses
Android Beam, an NFC-based communication channel that is created between two devices that
are within a few inches of each other. Android Beam creates a channel that is short-lived and
private to facilitate exchange of sensitive information during device pairing. The second method
displays a two-dimensional barcode on the screen of a device and the user scans that barcode with
the other device. This mechanism is also sufficiently secure to exchange sensitive system setup
information.
In-band policy management was implemented using digital signatures to authenticate policy changes. When the parent wishes to change the access control policy on a child device, a special capability is created by the parent and digitally signed. The digital signature was implemented with 1024 bit RSA and the MD5 hash algorithm.

Remote location tracking is implemented by making PMC Child a passive listener to Android device location changes. This means that PMC Child does not consume resources, such as battery, to monitor the child device location. When a remote tracking request is received from a parent device or an emergency notification is sent from the child device, the child device will attempt to obtain an exact location. If a location cannot be obtained quickly, the last recorded location will be sent. The location sent to the parent device is always accompanied by a measurement time and an error ellipse.

Remote lock is implemented using the Android Device Administration API. When the child device is paired with a parent device, the user is asked to enable PMC Child as a device administrator. The Device Administration API allows an application to lock the screen. This functionality is used by PMC to perform remote locking. Device Administration also protects the PMC Child App from being terminated or uninstalled by the child. This means that the ability to contact a parent or an emergency number is not provided by the prototype.
In this chapter the results of the PMC prototype are described and discussed. The system was deployed to Android smartphones and tablets. The Child App and AOSP build was deployed to Nexus 4, Nexus 5, and Nexus 6 smartphones and Nexus 10 tablets. The Parent App was deployed to various smartphones, including the aforementioned Nexus smartphones and Samsung Galaxy S3, Galaxy S5 and Galaxy S7 smartphones.

7.1 System Setup

The PMC system setup procedure begins with a parent device. The parent first creates a device family, as shown in Figure 7-1. The initial PMC Parent screen presents the user with two options (a). The first option is to create a device family. When this option is selected, the user is presented with a screen that collects a name for the device family, a name for the parent who will use the parent device, and an optional name for the device (b). After the Confirm button is pressed, the device dashboard is displayed (c). At this point there is only one device in the device family, the parent device used to create it. The second option that is presented to the user on the initial PMC Parent App screen is to join an existing device family. This is the option a parent selects to add another parent to an existing device family.
Once a device family exists, devices may be added to the device family. Figure 7-2 shows the process of joining a child device to the device family. When the Child App is run on the child device, the user is presented with two options for joining an existing device family (a). Note that child device cannot create a device family. The first option is to join using NFC. The second option is to join using barcode scanning. In Figure 7-2 (a), the user selects Join Family via NFC on the child device. The child device screen instructs the user to hold the devices together (b). When the devices connect via NFC, the system setup information is transmitted from the child to the parent. The Parent App then displays a dialog to collect a child name, icon and an optional device name (c). After the user presses the Confirm button, the PMC Parent App instructs them to touch the devices together again (d). When the devices connect via NFC, the PMC Parent App transmits the system setup information to the Child App. The Parent App then adds the new device to the Device Dashboard (e), and the Child App displays a message indicating that the devices were joined successfully (f). The Child App instructs the user to enable PMC Child as a device administrator and displays the Activate Device Administration screen for PMC Child (g).
Remote device release allows a parent to dismiss a device from the device family, returning the device to normal. To release a device, the parent uses a screen in the Parent App. The child device receives the notification shown in Figure 7-3. Parents can also release other parent devices from the system. When all devices have been dismissed from the device family, the last parent device can be reset using the Parent App. A parent can also reset the entire system, dissolving the device family and releasing all devices from the Parent App.
7.3 Local Mediation

The mediated operations from Table 5-2 are mediated according the policy stored on the child device. Each time the child performs a mediated operation, the local policy is checked. If the operation is allowed, there is no interruption to the child’s workflow. When an SMS operation is not allowed, it is blocked and a notification is displayed to the child as shown in Figure 7-4 (a). At the same time, the child is presented with an option to request permission to perform the blocked SMS operation. The Android Messaging app indicates that the message was not sent (b). At any time, the child may run the he PMC Child App. The Child App displays a history of all denials with status information (c). The Child can also request policy changes from within the PMC Child App. Figure 7-5 shows the denial of an attempt to place a call to an unapproved number (a), and the message displayed by the Android Phone app (b). Figure 7-6 shows the denial of an attempt to install an unapproved app. The app is shown being downloaded in (a), the PMC notification appears in (b), and the notification generated by the package manager is shown in (c).
Figure 7-4. Local Denial of an SMS Send Operation

Figure 7-5. Local Denial of a Place Call Operation
7.4 In-band Policy Management

In-band policy management allows the child to request changes to denied operations and get immediate responses from a parent. As just described, when a mediated operation is denied, the child is presented with a notification informing them of the denial, and allowing the child to request permission to perform the operation. When the child requests permission, the parents are immediately notified on their devices. The Parent App displays a notification. For SMS messaging and phone call approvals, the child will immediately be able to accept and receive messages and calls. For app installs, the Child App automatically installs the application. The screenshots in Figure 7-7 show examples of the In-band policy management sequence for requesting permission to install an app. After the child requests permission to install an app, the parent receives a notification on the parent device (a). The Parent App displays the pending request (b) and (c). The parent can view details of the request (d). If desired, the parent can view the app in the Play Store by pressing the provided link (e). When the parent approves the request, it is removed from the Pending Requests list and moved to the History tab (f). The child device immediately receives a notification of the approval (g) and the app is automatically installed (h).
Remote policy management allows the parent to make policy changes without a request from the child. The parent can manage white and black lists for SMS and phone calls, lock and unlock apps, setup device usage schedules, and remotely lock and unlock the child device. The screens for managing SMS and phone call white and black lists are shown in Figure 7-8.
Figure 7-8. SMS/Call White and Black Lists

The parent can remotely lock and unlock apps on the child device using the App Management screen shown in Figure 7-9 (a) and (b). When the parent locks an app, the child device receives a notification, as shown in (c). If the app is currently running on the child device, it is shut down. The child will be unable to start a locked app until the parent unlocks it from the parent device. In (d) the locked app icon on the Parent App Management screen displays a lock annotation indicating that the app is locked.

Figure 7-9. Remote App Locking
Remote locking and unlocking of the device is completed from the Parent App using the screen shown in Figure 7-10. When the device is unlocked, the unlocked icon is shown as in (a). When the parent taps the lock icon, they may enter a lock delay and the duration for which the child device will lock (b). When the delay expires, the child device will lock. When the duration expires, the device will automatically unlock. The child device can also be unlocked manually at any time. To do this, the parent taps the locked icon (c).

![Figure 7-10. Remote Device Lock and Unlock](image)

When the child device receives the lock command from a parent, a message is displayed, as shown in Figure 7-11 (a), indicating when the device will lock. Once the lock delay, if any, expires, the screen will lock and the child will not be able to unlock the screen. The phone will display a message on the lock screen as shown in Figure 7-11 (b).
Device usage schedules can be setup from the Parent App. The screens shown in Figure 7-12 allow the parent to specify multiple time intervals when the child device will automatically lock.
7.6 Remote Location Tracking

Remote location tracking allows the parent to obtain the current location of the child device, when it is available. The Parent App provides a screen for requesting the current location of a child device, as shown in Figure 7-13. When a location is returned to the Parent App, it is displayed on a map along with the time the location was measured and an error radius.

Figure 7-13. Remote Location Tracking

7.7 Emergency Notification

Emergency notifications are initiated by the child using a screen on the Child App shown in Figure 7-14 (a). When the child sends an emergency notification, the device obtains its current location and sends a message to the parents. Each parent receives an immediate notification on their device, shown in (b). Tapping the notification brings up the Parent App and displays the location reported by the child device on a map just like when a location is requested by the parent (see Figure 7-13).
Parents are able to remotely uninstall apps from child devices using a screen in the Parent App as shown in Figure 7-15 (a). When the parent chooses to uninstall an app from a child device, the child device displays a notification to the child (b), and the app is then uninstalled from the device. Not implemented with the prototype is the ability to remotely install apps on the child device from the parent device.
The PMC prototype implements the functional requirements consistently with the system design principles. It provides on-demand parental control using the internet connectivity of mobile devices. The parents are able to administer the child devices and children are able to request policy changes without the need to be physically present with the parent. The child devices locally enforce the policy set by the parents. The in-band policy management is fine-grained yet simple and easy to understand. It allows the parent to administer the use of the child device with no false positives or false negatives. As the child encounters access denials, they are easily able to request updates to the access control policy from the parents. The parents are able to receive and respond to access control requests immediately, regardless of their physical proximity to the child device. The setup of the system is simple, secure and performed in an intuitive way, by placing the devices together or scanning one device with the other. The system functionality is implemented with system framework hooks and a system service to guarantee system availability in the presence of the threats identified in our threat model. This design also allows PMC to completely mediate sensitive operations. PMC communications are protected using encryption.
and no sensitive information is ever exposed to a third-party channel. The system also guarantees authentic messages and policy change capabilities using digital signatures.
Chapter 8

Future Work

While PMC is a functional proof-of-concept parental control system, there are several areas to be addressed by future work. One weakness of the system is that it is only able to mediate operations that pass through the system framework in an identifiable way. There are many messaging and social media applications that PMC cannot easily mediate because the system framework is used in a generic way to send data to third-party servers. It would be impractical to mediate operations in general third-party applications. This challenge requires more research. In the meantime, PMC allows parents to control which apps are available to the child for use and to control what apps are installed on the child device. Child-friendly browsers and similar third-party apps could be installed on the child device and the unmediated system counterparts disabled using remote app management.

Since PMC is designed using system hooks, it cannot be deployed as a typical third-party app. PMC could be developed by a system integrator who is already enhancing the mobile operating system. This is the case with many device manufacturers for Android devices, and also with Apple and Microsoft. Third-party framework developers, like CyanogenMod, could also integrate PMC into their offerings. Another possibility would be to work with the AOSP community to integrate mediation hooks into mobile operating systems in a way that allows something like Device Administration to extend to parental control use cases.

While included in the system design, the current implementation of remote device locking does not allow emergency calls to be made while the device is locked. Instead of using the screen locking capability provided by Android Device Administration, an approach is needed that allows emergency calls to be placed regardless of the system lock state.
The PMC system user interfaces are sufficient for research, but require work before they are ready to be offered to the public. User interface improvements and polishing is a never-ending task for any application.

Many additional features could be added to the PMC system. Language filtering in text messages, detection of cyber-bullying, location alerts, and interfaces for family chat to name a few.
Chapter 9

Conclusion

PMC is a new approach to parental control systems that provides usable fine-grained parental control in a secure and privacy-respecting way. PMC is enabled by the notions of ubiquitous computing and high connectivity which enable in-band policy management. PMC employs classical security principles to provide a secure and privacy-respecting implementation. It is based on a simple, intuitive approach to policy management that is natural, fine-grained and effective, avoiding unintentional denials. PMC provides complete mediation of sensitive operations including phone calls, SMS messaging, app install/uninstall, app use and device use. In addition, PMC provides several industry-standard parental control features, like location tracking, emergency notifications, and SafeSearch. PMC is fully mobile, allowing parents to control child devices from their own smartphones without the need for physical access to the child’s device. User privacy is preserved using end-point encryption with a secure system setup procedure that prevents unencrypted data from traversing public channels such as the internet. PMC Child App and modified operating system was implemented using the Android Open Source Project, version 5.0.1, and tested using Nexus 4, 5, and 6 smartphones and Nexus 10 tablets. The PMC Parent App runs on commodity Android operating systems and was tested on Galaxy S3 and Galaxy S5 smartphones running Android OS versions from Ice Cream Sandwich through Marshmallow.
References


