Table of Contents

Chapter 1

Chapter Heading

1.1 Section Heading

1.1.1 SubSection Heading

SubSubSection Heading

Paragraph Heading with some text after

- Itemize
- One
- Two

1. Enumerate
2. One
3. Two

Description Description

One First

Two Second

Table	Headers	here
Lots of	rows	go here
Right Aligned	Left aligned	Centered

Table	Paragraph
Small header	lots of writing that can turn into a paragraph and will take up all of the remaining space in the page.
This	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

[?]
[?]
[?]
[?]
[?]

This is a sentence in green
This is in red
This should be blue.

'\\' above means `newline`. Repeated white space is always treated as a single whitespace—space, tab, newline.

Paragraphs are separated by a blank line.

- `Typewritter font`
- *Emphasised is better than bold/italics with* emph *inside*
- **Bold font with *emph inside***

```
public void javaMethod() {
   doSomething();
   if (happy) {
      clap(hands);
   }
   return new String("Really want to show it.");
}
```

Listing 1.1: This is the caption

Chapter 2

Introduction

2.1 Motivation

As the popularity of mobile communications devices increases, there is a growing tendency to use these as a convenient means of reviewing and revising documents on-the-move. Where these documents are of a confidential nature, particular attention must be paid to the fact that mobile devices are more vulnerable to compromise that traditional desktops, which are usually more extensively protected by the security measures implemented as part of an organization's internal network.

There are multiple mechanisms for keeping files secure on company servers whilst allowing employees the necessary permissions to work collaboratively with sensitive data as required. As the mobile device culture becomes more prevalent in the workplace, the addition of Mobile Device Management (MDM) applications empowers users to also access corporate data via their mobile devices whilst still allowing IT departments to retain a degree of control over data security.

Thus, it is acknowledged that maintaining the security of confidential documents can be challenging, even with the weight of a corporate IT infrastructure behind it. In this project, we seek to address the issue of allowing groups of users from different organizations (i.e. with no shared IT infrastructure) to collaborate securely on confidential documents and furthermore, to access these documents via a smartphone or tablet computer whilst minimizing the risk of exposing sensitive information to a potential attacker.

2.2 Project Aims

The primary aim of this project was to implement a scheme for the secure sharing of confidential documents between small (typically ¡ 15) groups of collaborators, subject to the following constraints:-

- Groups are self-organizing and represent multiple organizations, hence they cannot draw on the support of any central IT services.

- The documents involved are confidential in nature and hence should be encrypted both in transit and at rest.
- Group members wish to be able to access documents on a mobile device which is running the Android operating system.
- The solution devised should use only well-tested cryptographic techniques and standard libraries and should minimize the amount of trust to be placed in a third-party.

In pursuit of these aims we developed a solution called Securely Share, consisting of a detailed design of the security components of the system and a prototype android application (SecurelyShare) to provide a platform on which to implement and evaluate the various security features. It was acknowledged that, in a live setting, documents would usually originate on a PC rather than on a tablet device and thus the system would also need to a PC-based component. However, within the time constraints of the project it was considered infeasible to develop a fully featured system; our solution is submitted rather as a 'proof of concept'.

2.3 Overview of Report

The subsequent chapters of this report will deal with the design, implementation and evaluation of the project. Chapter 2 introduces some of the background material on key technologies used and presents an overview of the android applications reviewed as part of our preliminary research. In the light of this research, Chapter 3 presents a detailed analysis of the problem and expands the aims outlined in TODO into a more complete project specification, including details of the threat model against which we are attempting to defend. Chapters 3 and 4 deal with the solution design and implementation Need detail of rest of chapters here

Chapter 3

Background Material

In this chapter we will introduce some key aspects of the android architecture and its security features. We will also examine the features offered by the Dropbox API and finally we will look briefly at some of the commercial applications which were reviewed as part of our initial research and which offer some features similar to Securely Share.

3.1 Android Architecture and Security Features

3.1.1 Android Components

3.1.2 Android Permissions

ANDROID COMPONENTS ANDROID PERMISSIONS

3.2 Dropbox API

Dropbox is a cloud storage service that also offers users automatic backup facilities, file synchronization across devices, and the ability to share files with other users. It provides multi-platform client applications plus a series of public APIs that enable different subsets of the Dropbox functionality to be integrated into third-party applications.

On the Android platform, Dropbox offers three APIs:

Core The Core API provides a flexible way to read and write to Dropbox. It includes support for advanced functionality like search, revisions, and restoring files. While Drop-ins and the Sync API are simpler to integrate and use, the Core API can be a better fit for server-based apps.

Datastore With the Datastore API, structured data like contacts, to-do items, and game state can be synced effortlessly across devices.

Sync The Sync API is like having a private filesystem for your app with its very own Dropbox client running in the background. Your app can read, create, and modify files using familiar filesystem-like calls. Behind the scenes, the Sync API takes care of syncing changes to Dropbox and notifying your app when changes are made in Dropbox, so you can update your UI instantly.

For the purposes of the SecurelyShare app, the Dropbox Sync API was deemed most appropriate.

When you start building an app on the Dropbox Platform, you'll need to create a Dropbox app in the App Console. As part of the process, you'll need to choose the right permission for your app. Your app's permission (sometimes referred to as access type in the documentation) determines what data your app can access in a user's Dropbox. For the Sync API, three levels of permission were applicable:

- App folder
- File type
- Full Dropbox

TPM? REVIEW OF OTHER AVAILABLE APPLICATIONS "http://source.android.com/devices/tec
and-kernel-level-security"

"http://link.springer.com/book/10.1007/978-1-4302-4063-1page-1"

Chapter 4

Analysis and Specification

Key generation and sharing currently done from PC rather than android device Folder management and sharing done outside of app - could be added but may require switching to a more complex dropbox api. Cost-benefit analysis

Early design considerations • who generates keys • how are keys generated • how are keys distributed • issue with public key - how would we stop Mallory uploading bogus documents • issue with where to place trust • how to manage letting decryption know group - flirted with shared preferences Issue: if generate [rivate key on device, it is device specific - ability to import would allow same keystores to be used on multiple devices for4 same user • decision to ignore considerations like battery life and User authentication and need to block after failed attempts • large files • network connectivity • battery life • small memory • multithreading for gui • where to encrypt • model to use for file distribution and storage Means of distributing certificates is outside of scope - as these are public, any means will do.

Chapter 5

Solution Design

Reasons why didn't choose ID based cryptography or password based solution - aim is to use simplest solution that works

Design. A high-level account of the structure of your software and how it works. What algorithms does it use? How do these compare with alternatives? What were the main design decisions you took, and their justifications?

Althou prototype allows for encryption of files stored on the device, in practice the very fact that there are files on the device that the user wants to encrypt violates our central tenet that plaintext should never be stored to disc.

Chapter 6

Implementation

Implementation and testing. A detailed account of the implementation and testing of your software. Explain what data structures you used, and how the algorithms were implemented. What implementation decisions did you take, and why? There is no need to list every little function and procedure and explain its working in elaborate detail; use your judgement on what is appropriate to include. use of .xps, .xeb For improvement, use custom file extension registered with Dropbox then would only ever see encrypted files use of bundle for passing data between activities use of interface for passing data back from dialog Performance problem - introduced buffering Splash screen and initialization didn't use onStoreState etc. - didn't worry about restoring exact user position as prototype and system stores GUI stuff Use of singleton Keystores moved to external storage for testing and demonstration purposes. In a production app, these should be moved back to internal storage in order to take advantage of the additional protection afforded by android's inbuilt security mechanisms Removal of keystores upon 3 successive failed password attempts – al present it just shows a message saying that the keystores have been deleted but doesn't actually remove them from the device. In a live system this would need to be implemented. • use of xml rather than java for managing onClick - why was this done and when is it not applicable Major issues with keystore, certificates and default providers. Challenge of absence of built in file manager DROPBOX ISSUES • dbx stuff does not implement serialisable or parcelable • Dropbox synchronization issues - developed everything using App specific access then discovered that this doesn't allow any use of shared folders so had to redesign Challenge of unavailability of BKS on pcs in school No access to key tool in android Include information about algorithms and key lengths Fragments TESTING Testing - it is ok to say that I tested by inspection Explain why unit testing is not meaningful

Issue: if generate [private key on device, it is device specific - ability to import would allow same keystores to be used on multiple devices for4 same user

Assumption that encrypted blobs are probably also created on PC - simple PC version of program developed to address this, although no gui developed

Chapter 7

Evaluation

ACHIEVEMENTS • What works well EXPLAIN HOW WELL SOLUTION MEETS OBJECTIVES - WHAT YOU HAVE LEARNED - WHY ANDROID DEVELOPMENT WAS A CHALLENGE major challenge of the fact that android is an operating system not a programming language - event driven programming FURTHER WORK • From prototype to production - next steps Write as though you are porviding a basisi for a good cs graduate to continue the work - assume they have already done some android development SECURITY EVALUATION Did not implement signing in prototype as largely meaningless with self-signed certificates. Purchase of appropriate certificates for authentication of signatures would be required for a complete solution Attacks and issues to consider • anonymity • forward secrecy • revokation • man-in-middle Delete keys after failed password attempts Write about how p[rotocols as as important as implementation - need to support this view from academic papers Talk about why it doesn't matter that encrypted copies of group key are available on dropbox nelenkov.blogspot.co.uk - credential storage enhancements in Android 4.3 out of bounds channel - side channel attack Write about issues to do with public key distribution and the need for signing Talk about decision not to implemenmt passing decrypted data directly to another app without needing to write to external storage Don't zero out passwords after use No implementation of digital signatures so vulnerable to man-in-middle Decision to use same password for keystore and aliases - trade off of added security against temptation for users to use insecure passwords or write them down

Group needs admin, although any group member can serve in this role. It is also possible to delegate this to an administrator who is not part of the group without giving them access to the group encryption key. However, if the admin was corrupt, the fact that they had access to the private key for signing the encrypted group key would still be a problem.. Useful phrase "a more sophisticated attacker"

Threats: • Malware on device • Attacker snooping around external storage but not one with root access • Lost device with app open (minimal protection) but can unlink from dropbox remotely so would only have a very small window of opportunity to decrypt files currently stored on devicewhilst keystore is unlocked • Could have had different password for each group • Could make user re-enter

password for each file – trade off between added security in event of lost device and templtation for user to choose a weaker password

Maybe argue why solution is secure here PROTOTYPE EVALUATION dependent on exactly correct alias for groupid and folder name • Is designed as a "proof of concept" • Aspires to use "best practice" within the code • Uses well-tested cryptographic techniques and standard libraries • Adheres to the stated security requirements No ability to change passwords etc added at present

EVALUATION OF PERSONAL LEARNING • zero knowledge starting point • Android is a whole new operating system not just 'Java with extra bits' • Unfamiliar API's operating in a sub-optimal environment