An Android PGP Manager: Towards Bridging End-User Cryptography to Smart Phones

Benjamin Aziz *benjamin.aziz@port.ac.uk*
School of Computing
University of Portsmouth
Portsmouth, PO1 3HE, United Kingdmon

Mario Tejedor-Gonzalez *1077@usal.es*
F.A.C. Nhorte, S.L.
Finca los Malatos
Arguero-Villaviciosa 33314
Asturias, Spain

Abstract

This paper presents a key management and an encryption application for the Android operating system, which can be used to secure the privacy of messages exchanged among applications and Web services when accessed from mobile devices. In addition to the management of PGP keys, the application uses PGP encryption to secure the clipboard communication channel on the Android OS, which is widely used to exchange messages especially in the context of social networking and other Web-based services.

Keywords: Mobile platforms, Security, Key Management.

1. INTRODUCTION

The great rise in ubiquitous computing in recent years has provided great opportunities for information flow and mobile computations to facilitate the concepts of “business on the move” to become an essential part of our day-to-day activities. It is be-coming part of the norm that our personal and business data flows through millions of mobile devices and computers every day, being copied, inspected and altered thousands of times in the process leading in most cases to various degrees of mutations of the original data. The treatment of this data is often complex and expensive, which discourages information service providers from guaranteeing its security and the end-user privacy, and prefer to keep their commitment at a very low profile. Therefore, it is often the case that the final responsibility for keeping sensitive information private and secure relies on the end-user themselves who should be aware of the services’ terms and conditions before they commit to their usage, often a hard task given the high number of services a user may use each day.

Throughout the years, the research community has developed algorithms to improve user data privacy, among other security concerns. In 1997, the Internet Engineering Task Force (IETF) [1] was proposed to build a public standard for Pretty Good Privacy (PGP) [2], which led the private solutions to be able to reach compatibility with each other and let the private sector start developing open solutions for signing and encrypting data. The OpenPGP standard was suggested by the IETF in 2007 [3] in order to transform PGP into a public standard that can be implemented in various ways. This standard has been widely adopted since then by many developers and companies and it has been implemented in several software applications and protocols (e.g. SSH protocol, Gnu Privacy Guard Application, PGP Corporation etc., see [18] for a full list of its members).

The idea of implementing cryptography on handheld devices is a common idea that has been investigated ever since computing went mobile. For example, in [17], Tillich and Großschädl

consider the feasibility of implementing cryptographic algorithms on the old form of handheld devices that were Java-enabled predating the current OS-enabled generation.

This paper follows similar footsteps in adopting the OpenPGP standard for developing a cryptographic solution compatible with the widest range of smart phone soft-ware possible using the Android operating system. The solution aims at providing privacy of data through the encryption of the phones clipboard as well as the management of the users' encryption keys. The clipboard is an easy means of communication across services and often holds transient data that may carry sensitive information related to the user's privacy and security.

Smart phones themselves have evolved so much that now they are capable of car-rying out the most complex tasks that were only meant to be done by desktop or laptop computers a few years ago. Therefore, smart phones have become one of the most demanded platforms for the development of new applications and information services. This quick spreading of services around smart phones has brought a huge variety of end-user services each tagged with different legal clauses and restrictions, sometimes without much assurance for the user's privacy or the security of their data.

As these devices usually need a network connection to use such services, many connection services have been spread around the globe, each one with different privacy conditions and commitments. A regular smart phone user uses several services from different companies a day, which is difficult to keep track of when it comes to legal issues relating to information's privacy.
The rest of the paper is structured as follows. In Section 2, we give an overview of other solutions that have been proposed for using cryptography in security mobile information, and discuss their pros and cons. In Section 3, we discuss the require-ments and use cases motivating our solution. In Section 4, we give an overview of the architectural design of the PGPM application, and in Section 5, we discuss our implementation through the various user interaction views. Finally, in Section 6 we evaluate the application against the original requirements and conclude in Section 7.

2. TECHNOLOGY REVIEW

In this section, we discuss some of the relevant technologies currently deployed in the market to provide cryptographic solutions to mobile devices as well as some existing mobile platforms and other technologies relevant to the PGPM application proposed in the paper.

2.1 OpenPGP-Compliant Applications

There are various OpenPGP-compliant applications designed for different platforms. We focus here particularly on the functionality and capabilities of such applications to understand what a smart phone user could expect from this kind of software.

KGPG for Linux. The first application reviewed is KGPG for the Linux platform [4], which operates under the KDE Desktop environment. This application provides a Graphical User Interface (GUI) for users to create and manage key-pairs, as shown in Figure 1.

It also provides functionality for importing/exporting key-pairs from/to a key server. KGPG runs as a plugin to a mail client and can act as a text-encryption/decryption or text-signing/verifying service. The plugin can also be used from a file browser as a file-encryption/decryption service. However, there are a few disadvantages to the application. Mainly, it lacks an interface that provides file and text encryption. Additionally, the digital signature functionality cannot be used outside a mail client. Finally, it lacks compatibility with applications that provide no plugin support.

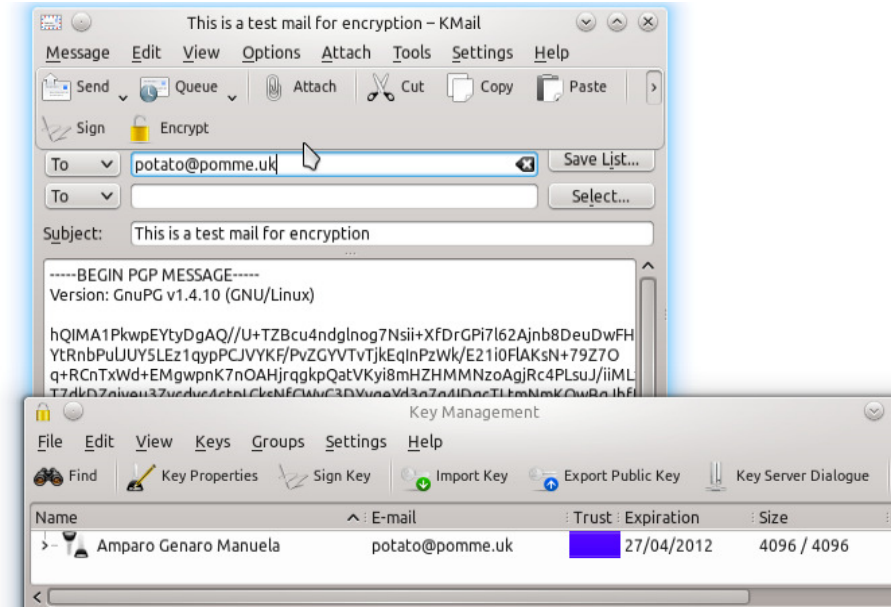

FIGURE 1: KPGP for Linux.

GPGMail for OSx. The second application reviewed is GPGMail [5] for the OSx platform [6]. This application provides a plugin for OSx's mail client, which can be used to add capabilities such as key-server integration, as shown in Figure 2.

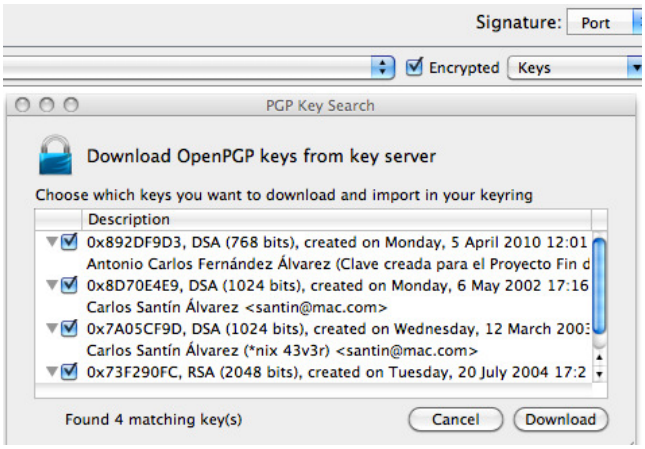

FIGURE 2: GPGMail for OSx.

Additionally, mail encryption/decryption and mail signing/verification can be per-formed as well as offering PGP S/MIME attached signatures compatibility. The main disadvantages of the application are that it is not a standalone GUI, it has no file-encryption functionality and it is not compatible with other applications.

Gnu Privacy Guard (GPG). The third and last application reviewed was the Unix-like command-line tool called Gnu Privacy Guard (GPG) [7]. As this piece of software does not implement a graphical interface, its maintainability seems to be less complicated, so it has more features than the previous two applications and it has also been ported to Windows/Linux/Unix/OSx/BSD systems. The main features of GPG include: Key-pair creation and management, key-pair import and export, key-server integration, text encryption/decryption, text signing/verifying, file encryption/decryption and file signing/verifying. However, it has two main disadvantages: There is no GUI and hence, it requires user knowledge in command-line usage.

2.2 Mobile Platforms

Smart phones market has a wide variety of platforms to develop software for that is being increased every year due to the evolving nature of this young field of research. We discuss below the two currently most popular platforms, iOS and Android.

iOS. iOS [8] is an Operating System for Apple devices that supports bundled applications. A bundled application is a piece of software that is never installed into the OS itself, but is kept in a safe and restricted environment called 'sandbox', which will not allow the application to modify the standard behavior of the OS while it is executing. This system allows applications to communicate with the OS and use its services, such as sending an email, looking for a contact in the address book and displaying the GUI. These restrictions provide extra security at execution time because no application can interfere with another, but make the OS impossible to be extended by another than the OS developers. No file encryption can be supported due to the lack of access to the file-system. Furthermore, no mail decryption is supported due to the impossibility of modifying the mail client and due to the legal restrictions on re-implementing core functions such as the mail client program.

Android. Android [9] is another OS that supports bundled applications but in a more open-minded way. The Android platform consists of a modified Java™ Virtual Ma-chine [10] running over a modified Linux kernel. Such modifications were applied to the original project to achieve a better performance of those systems working together in a smart phone device, which has limited resources. The Android's Application Programming Interface (API) is based on the basic Java environment, however supplied with Android's libraries to make it even more powerful.

The Android OS works with different hardware platforms: HTC Smart phones, Samsung Smart phones, Digital Video Broadcasting receivers, Tablet PCs etc. It also keeps applications in sandboxes, but these applications can communicate with each other, launch each other and pass data to each other when prepared. This keeps a good level of security and also offers flexibility when it comes to making one application available to be used and complemented by others. Also, even when Android OS does not allow any applications to modify its default behavior, it still provides the developer many ways of letting the user tune the OS to gain interactivity. For example, the user can select which program should be opened to view an image or to edit a text file, or which action should be performed when a certain type of a file has been downloaded. This could be extended and developed as to let an application ask for a program that can provide encryption or decryption and let another program answer that call offering encryption/decryption services.

2.3 Other Technologies

2.4

Bouncy Castle. In our paper, we shall use the Bouncy Castle [12] cryptography libraries to provide the cryptographic functionality in the application. These libraries are based on Java, which makes them fully portable to a wide range of devices.

SQLite. The keys will be stored using a lightweight storage system called SQLite3 [13], which provides structured queries support when retrieving and inserting data. It behaves like a database server, but it is not. It has no server-client structure. It is just a piece of code that can be included and ported to a huge variety of devices and projects due to its small size and its high performance. The whole database structure and data can be stored in one file that also can be ported and used in multiple platforms due to its inter-platform format. However, as Jay A.K. points out in [14, page 10], SQLite is not a database system, it does not provide query optimization, so a higher knowledge about databases may be needed when dealing with advanced queries in order not to waste processing power or other resources.

Although SQLite is not designed to support the level of concurrent access provided by many client/server relational database management systems, it will serve to our application's purposes

since the database is only going to be accessed by one execution thread from one process, the application, at the same time, hence no concurrency problems need to be resolved.

Linux VFS. Android is a Linux-based platform, which means that there is a Linux-based kernel running on every Android device. The Linux Virtual File System (VFS) is known to be a very safe and flexible system that controls the access to every file and folder on the system under a policy that attends to users, groups, and the public, giving each set a different level of access to files and folders.

Android assigns a different User ID and Group ID to each application installed on the system, preventing them from accessing one another's data if no further action is taken. If there is a need for sharing application data across applications, Android can arrange a tuning of the group permissions for two or more applications that need to share certain application data files (see [19]).

2.5 Conclusion on the Technology Review

In order to define the specifications of our application, it is important to decide on a suitable platform that supports and provides enough services to accomplish the requirements of our application as discussed in the next section, and to allow the development of future desirable features that were not implemented in the current version of the application. Therefore, it is mandatory to keep in mind that the platform used needs to have easy and good quality maintenance to provide the user with the best experience possible.

The strategy we followed was to take a look at the usage statistics for the various platforms in order to choose a platform that can offer the application to the widest range of users. Hence, as can be seen from Figure 3, according to comScore Mo-biLens surveys [20], the three most dominant platforms in the U.S. market were Google's Android, Apple's iOS and BlackBerry's RIM [15].

Top Smartphone Platforms
3 Month Avg. Ending Nov. 2011 vs. 3 Month Avg. Ending Aug. 2011
Total U.S. Smartphone Subscribers Ages 13+
Source: comScore MobiLens

	Share (%) of Smartphone Subscribers		
	Aug-11	Nov-11	Point Change
Total Smartphone Subscribers	*100.0%*	*100.0%*	*N/A*
Google	43.8%	46.9%	3.1
Apple	27.3%	28.7%	1.4
RIM	19.7%	16.6%	-3.1
Microsoft	5.7%	5.2%	-0.5
Symbian	1.8%	1.5%	-0.3

FIGURE 3: Size of the Smartphone Operating Systems Market in the USA.

After reviewing these platforms, Android was chosen due to the following reasons:

- Android provides access to the file-system, which brings the opportunity of implementing file encryption and basic key-pair management.
- Android allows applications to interact with each other and run at the same time, so for example, it can create a communication channel between a mail client and our application thereby offering the user the option of signing/verifying emails.
- Android provides a fluent communication between two or more applications by transmitting all kind of data between them, which makes it possible to write an ap-

plication that integrates with the OS and all the installed applications instead of building every feature in the main application.

Moreover, we decided to design the application to run over the Android 2.2 OS as a minimum to ensure total compatibility with the current market. Android 2.2 OS is commonly used and provides many advantages and features to make the most of. Finally, since the Android's API is based on Java, our language of implementation was chosen to be Java with basic XML [11] for laying the graphical interfaces.

3. APPLICATION REQUIREMNTS

The main aim of the developed application is to provide a step towards bridging the gap in solving the privacy problem, which currently exists in mobile applications and Web services (e.g. Facebook or any other Web service) by securing communication channels between users when interacting with Web services using their mobile devices and mobile applications. In order to achieve this, the software must fulfill a few requirements related to the protection of data and provide correct functionality and desirable non-functional features. Most of these requirements are also inspired by the related technologies reviewed in the previous section.

Privacy: The application should keep text and files secure even though these are transmitted through non-trusted communication channels. For example, it should ensure that the recipient of the message sent over a Facebook page is the only one who can read it.

Integrity: The application should keep and verify information integrity. For example, it should verify that a message posted over Facebook is complete and unaltered.

Authentication: The software should be able to verify if the information comes from a trustable source and that the source of the information can be authenticated. For example, it should be able to check if the received message has been sent by the ac-count's owner and not by anyone on the owner's behalf.

Flexibility: The aim of the project is to improve as many other services possible, so it has to be compatible with as many applications and services as possible, for example, Facebook, Twitter, e-mail, web-mail and Web forums.

Compatibility: There are many privacy-based applications being used nowadays in many different platforms. The application should be compatible with most of them. In particular, it has to be compatible with related technologies such as GnuPrivacyGuard [7], KGPG [4] and GPGMail [5].

Efficiency: The application should run smoothly on the smart phone's platform, which effectively means that it should not use more than 5MB of the device's internal storage in order to keep a lightweight installation. The CPU should run at least at 300MHz to provide a smooth user experience.

4. USE CASES

Next, we discuss some of the use cases that impacted the design of the application presented in this paper. These mainly constitute its functional behaviour and its inter-action scenarios with the user.

Importing key-rings from a file: The first use case is related to the opening of a binary or a text file containing the key-rings used for the encryption by extracting the public or private keys that form the key pair and storing those keys on the device for a later use.

Exporting key-rings to a file: This use case will let the user transfer the already stored keys to a file in order to use those on another device or computer.

Managing key-rings: In this use case, the user will be able to see what keys are stored on the device and will be able of deleting keys that are no longer to be used.

Encrypting text to the clipboard: This use case will provide encryption to whatever content stored on the clipboard, so it will assure compatibility with lots of application and purposes.

Decrypting text from the clipboard: The user here will be able to decrypt content from the clipboard and read it if happens to be text.

Signing text on the clipboard: The user will be able to add a digital signature to whatever content stored on the clipboard, turning it into ASCII armored text content.

Verifying signed text from the clipboard: This will let the user check if the content on the clipboard remains unaltered and has been sent by a trustable source.

5. APPLICATION DESIGN AND ARCHITECURE

The design of the encryption management functionality in the PGPM application follows the Model-View-Controller (MVC) design pattern [21]. The Android OS does not strictly follow the MVC structure, however, one could use MVC to describe and model Android's flexibility. In MVC, every activity is a different controller, which instantiates classes from the model to use their methods, and process some data that will be sent to a selected view [19].

5.1 Activities

The application has two activities (or controllers) that will control the operations of the software: The main Home activity and the KeyList activity.

Home Activity: This activity is meant to start the application and prepare the home screen by plotting the main layout and creating the context menu for that view. The class definition for this activity is shown in Figure 4 below.

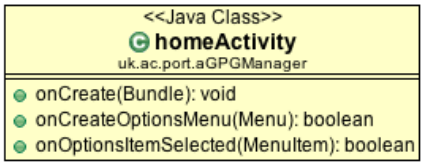

FIGURE 4: The homeActivity Class.

As in every Android Activity, this activity implements 'OnCreate()' method, which will be called when the application is launched. In order to create the context menu, which pops-up whenever the menu-key is pressed, and react to the activation of any of its elements, the activity also implements 'onCreateOptionsMenu()' and 'onOp-tionsItemSelected()', respectively.

KeyList Activity: This activity is the main controller of the application because it manages most of its functionality. As its name suggests, the associated view will display a list of public or secret keys and will react to the events raised by the GUI. As it can be seen in the definition of the class KeyList activity in Figure 5, it has some private variables that will be used to communicate different methods and actions launched from the current activity and will provide the activity with some state, which will determine the behavior of the controller itself. In order to run all the tasks the user can perform, 'mode' and 'action' properties will be updated from home Activity when it launches the KeyList activity.

The subclasses of the KeyList activity are procedures that will be called from another thread to interact with the platform while the main thread keeps the GUI updated. As it also can be seen from Figure 5, this class implements some event handlers to interact with the user input. (e.g.

OnListItemClick, OnOptionsItemSelected). In order to perform operations with the GUI, it will implement the methods showImportDia-log(), showSignItDialog(), showPassPhraseDialog(), which prompt the user to enter data that will update the state of the activity.

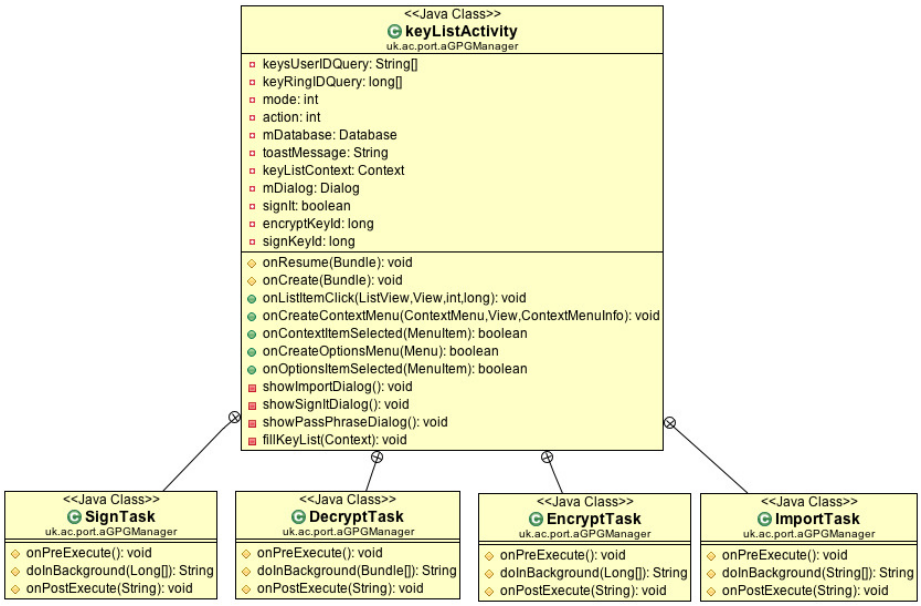

FIGURE 5: The KeyList Class.

5.2 The aGPGManager Database

The database structure and functionality were defined in a different class, called Database, shown in Figure 6(a). The class provides all the necessary methods to interact with the database and the data stored in it. It also implements some helpers that will take care of formatting special data types. The actual database is structured as shown in Figure 6(b).

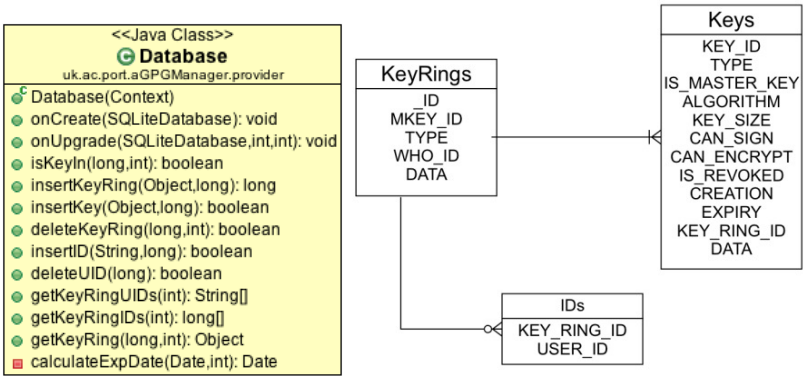

FIGURE 6: The aGPGManager Database Class (a) and its structure (b).

As the figure reflects, the database will consist of three tables: KeyRings, Keys, and Ids, which will store key-ring related information, the keys inside a key-ring, and the user identities for each key associated to its key-ring, respectively. The database was implemented as an SQLite database.

6. APPLICATION VIEWS

The design of the encryption management functionality in the PGPM application follows the Mo

In order to target the biggest number of devices, all the layouts were defined using relative measures and simple interfaces that were adapted to every screen without losing functionality. This provided a more general interface with less customization, but ensured t compatibility across different screen sizes and formats was maintained.

The first view that the user interacts with is the Main screen. As it is shown in Figure 7(a), the main screen is quite simple: it has two buttons to interact with the platform's clipboard, which gives the application compatibility with any text-based service running on the same device. By pressing the menu key, the menu will be deployed, giving the option to edit the keys on the database, as shown in Figure 7(b).

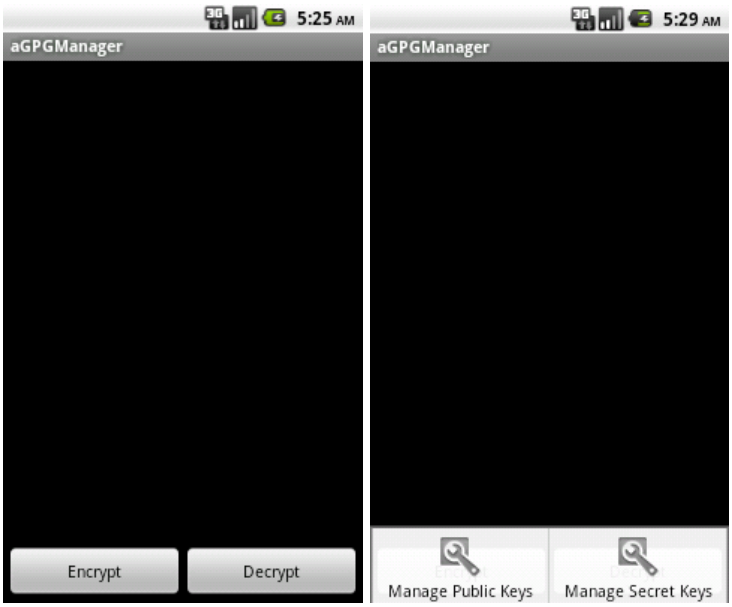

FIGURE 7: Main Screen (a) and Key Management Menu (b).

If any of the key managers is selected, the application will either manage the public or private keys in the database. Figure 8(a) shows the editor with all the available keys on the database, where the user can add more of these. If a key is pressed long enough, a new menu will pop-up asking for deletion confirmation as in Figure 8(b).

If the user needs to import a key, a dialog will pop-up prompting the user for the location of the file containing the key-ring, as shown in Figure 9(a). Every time the user performs a long-term action, bubble messages will pop-up to keep the user up-dated with the action's progress, as shown in Figure 9(b).

If from the Home screen (Figure 7(a)), the user taps on Encrypt, the application will ask the user to select a public key from the list to encrypt with, encrypting the clip-board as shown in Figure 9(a) above. If, on the other hand, the user taps on Decrypt from the Home screen (Figure 7(a)), the application will ask the user to pick a key to decrypt with and prompt the user for the passphrase, as shown in Figure 10(a). If the user copies text from any application (e.g. a note application), as shown in Figure 10(b), and performs the encryption tasks as explained in the above steps, then it is possible to view the encrypted contents of the clipboard by pasting them onto a new note, as shown in Figure 11(a). The outcome of the paste is the ciphertext corresponding to the contents of the clipboard, as in Figure 11(b).

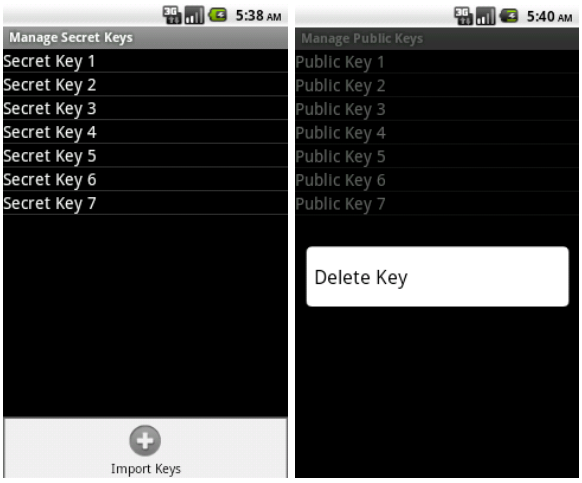

FIGURE 8: The Available keys in the Database (a) and the deletion pop-up (b).

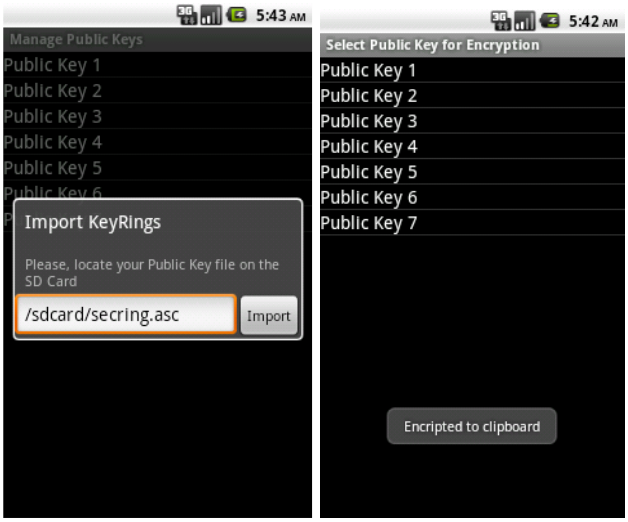

FIGURE 9: Importing Keys (a) and bubble information messages (b).

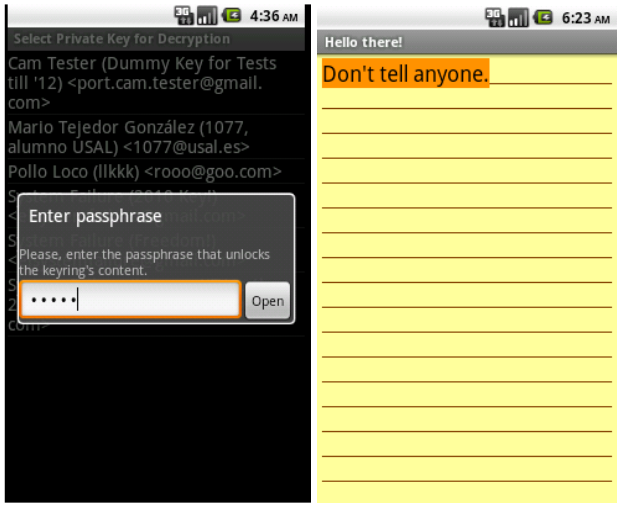

FIGURE 10: Importing Keys (a) and bubble information messages (b).

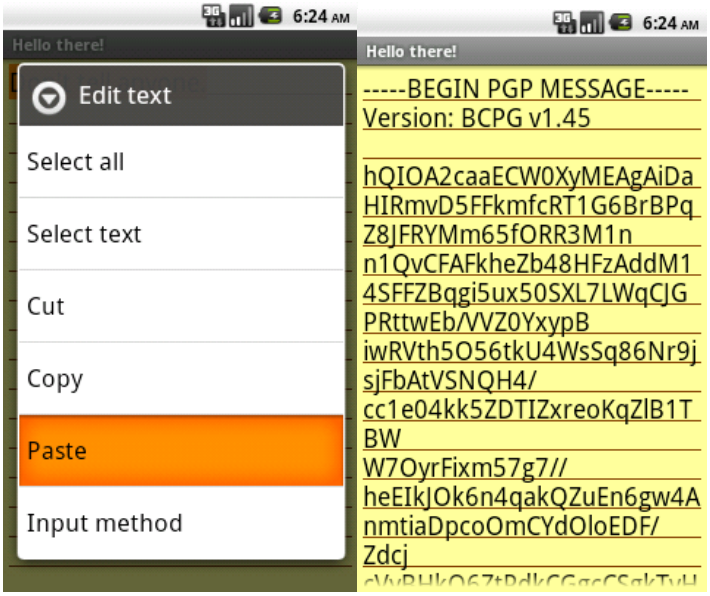

FIGURE 11: Pasting the clipboard (a) and the contents of the clipboard (b).

7. EVALUATION

We discuss in this section the evaluation of the PGPM application against the original requirements that were discussed in Section 3.

Privacy: The privacy requirement has been achieved through implementing clipboard encryption/decryption and is aided by the key-ring management feature of the application. This implies that whenever for example a user wants to post a secret message on a Web service application, such as Facebook or Twitter, they can conveniently do so by simply clicking a single button that encrypts the contents of the clipboard. Similarly, if in possession of the correct key, the user can decrypt an encrypted message simply by copying it and then performing a decryption of the clipboard content.

Integrity and Authentication: Both integrity and authentication remain areas for future improvement of the current version of the PGPM application. We believe that achieving this will be straightforward, since both rely on digital signatures, which can be implemented also using PGP encryption.

Flexibility: Clipboards provide a flexible and popular way of communicating information on mobile devices among applications and Web services, which is easy and cheap to use. In securing the use of the clipboard as a communication channel, we have provided a reasonably flexible solution to the problem of mobile communication privacy among such applications and services.

Compatibility: Our application is compatible with OpenPGP libraries, which is the open standard for the PGP cryptographic suite.

Efficiency: The performance constraints of the platform were adhered to, hence the PGPM application installs and uninstalls smoothly, runs without glitches and it up-dates user data as expected.

8. CONCLUSION AND FUTURE WORK

We have designed and implemented a PGP manager application for the Android OS platform, which helps bridging the gap in the use of cryptography on mobile devices when interacting with

applications and Web services, in order to protect privacy and secrecy of information. The application, based on the OpenPGP standard, allows users to manage their PGP key-rings and provides functionality for encrypting/decrypting the clipboard content, which is a commonly used and a quick channel of communications among applications and Web services. Future enhancements of the application will include supporting integrity and authenticity of the clipboard. There are other features that can also be designed and implemented in future versions of the application. These include:

- *Public-key distribution*: It would require basic network knowledge to implement key-server communication.
- *Autonomy*: It would be a matter of time to design and develop some procedures to create private and public keys from the Android platform itself.
- *File privacy with more flexibility*: Extending the application's interface to crypt and decrypt files would make it achieve the best compatibility possible.
- *Multilevel Security*: Implementing key-grouping into the design wouldn't be too hard. It could be done easily by adding a field called 'rank' to the key-rings table and implementing a policy of keys' order.

This kind of security would allow several people accessing the same data to retrieve just what is meant to be retrieved by each one according to their security rank (the number and order of keys they have to decrypt it).

9. REFERENCES

[1] Internet Engineering Task Force (IETF) – www.ietf.org

[2] Philip R. Zimmermann. The Official PGP Users Guide. The MIT Press, 1995.

[3] J. Callas, L. Donnerhacke, H. Finney, D. Shaw and R. Thayer. OpenPGP Message Format. Network Working Group RFC 4880, 2007.

[4] The KGPG Utility - http://utils.kde.org/projects/kgpg/

[5] GPGMail - http://www.gpgtools.org/gpgmail/index.html

[6] Apple OSx - http://www.apple.com/macosx/

[7] GNU Privacy Guard - http://www.gnupg.org/

[8] Apple iOS - http://www.apple.com/ios/

[9] Android - http://www.android.com/

[10] JavaTM VM - http://www.java.com/en/

[11] XML Specifications. W3C - http://www.w3.org/XML/Core/#Publications

[12] The Legion of the Bouncy Castle - http://www.bouncycastle.org/

[13] SQLite3 specification - http://php.net/manual/en/book.sqlite3.php

[14] Jay A. Kreibich. Using SQLite. O'Reilly Media, 2010.

[15] Blackberry RIM - http://us.blackberry.com/company.jsp

[16] V.A. Reston. U.S. Mobile Subscriber Market Share. comScore Press Section. Retrieved April 27, 2011, from http://www.comscore.com/Press_Events/Press_Releases/2011/4/comScore_Reports_February_2011_U.S._Mobile_Subscriber_Market_Share

[17] Stefan Tillich and Johann Großschädl . A Survey of Public-Key Cryptography on J2ME-Enabled Mobile Devices. In Lecture Notes in Computer Science, Springer, Volume 3280/2004, 935-944, 2004.

[18] The OpenPGP Members List - http://www.openpgp.org/members/

[19] Android Inc., Security and Permissions http://developer.android.com/guide/topics/security/security.html

[20] comScore MobiLens, http://www.comscore.com/Press_Events/Press_Releases/2011/12/comScore_Reports_November_2011_U.S._Mobile_Subscriber_Market_Share

[21] Hello Views Tutorial - http://developer.android.com/guide/tutorials/views/index.html