NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

TpHCM, ngày ….. tháng …… năm ……

Giáo viên hướng dẫn

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

TpHCM, ngày ….. tháng …… năm ……

Giáo viên phản biện

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Khoa Công nghệ Thông tin, trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em học tập và thực hiện đề tài tốt nghiệp này.

Chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy Trần Minh Triết, thầy Bùi Tấn Lộc đã tận tình hướng dẫn, chỉ bảo chúng em trong quá trình thực hiện đề tài.

Chúng em xin chân thành cám ơn quý Thầy Cô trong Khoa Công nghệ Thông tin đã tận tình giảng dạy, trang bị cho em những kiến thức quý báu trong những năm học vừa qua.

Chúng con xin chân thành cảm ơn Cha Mẹ đã luôn động viên, ủng hộ vật chất lẫn tinh thần trong suốt thời gian qua.

Chúng em xin cảm ơn sự quan tâm, giúp đỡ và ủng hộ của anh chị, bạn bè trong quá trình thực hiện khóa luận.

Mặc dù đã cố gắng hoàn thành khóa luận trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Tp. Hồ Chí Minh, tháng 07 năm 2008

Nhóm sinh viên thực hiện

Trương Toàn Thịnh – Nguyễn Đình Lê Hưng

MỤC LỤC

DANH MỤC CÁC HÌNH

DANH MỤC CÁC BẢNG

THUẬT NGỮ

Android	Hệ điều hành mới Google dành cho điện thoại di động
API	Application Program Interface, tập hợp các thư viện hàm dành cho phát triển ứng dung
Contact	Thông tin liên lạc của người dùng, bao gồm nhiều thông tin của nhiều người trên điện thoại
Contact Item	Thông tin liên lạc của một người dùng trên điện thoại
Content Provider	Kỹ thuật để truy xuất dữ liệu.
Dalvik	Mảo ảo Java trong Android, dùng để biên dịch mã bytecode sang mã thực thi, cò thể chạy được trên Android
Dalvik Execute	Tập tin thực thi chương trình.
GSM	Global System for Mobile Communication **(**hệ thống thông tin di động toàn cầu)
IDE	Integrated development environment – môi trường phát triển phần mềm.
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension – thư viện mã hóa mở rộng trong Java
Layout	Giao diện người dùng
Linux OS	Hệ điều hành Linux
SDK	Software Development Kit, công cụ danh cho phát triển ứng dụng.
SMS	Short Message Services – dịch vụ gởi tin nhắn, cho phép gởi các thông điệp dạng ngắn không quá 160 ký tự

TÓM TẮT

Ngày nay, điện thoại di động đã trở nên hết sức quen thuộc với mọi người, và ngày càng phát triển mạnh mẽ trên toàn thế giới.Ngoài những chức năng cơ bản gọi điện thoại, nhắn tin, điện thoại di động ngày càng tích hợp nhiều vai trò khác nhau như sổ tay ghi chú, lịch hẹn… theo đó, dữ liệu lưu trên điện thoại ngày càng gia tăng. Ngoài các dữ liệu thông thường như tin nhắn, thông tin liên lạc còn có nhiều thông tin quan trọng và nhạy cảm khác như thông tin về công ty, hợp đồng hay đối tác. Do dó, nhu cầu về ừng dụng bảo mật thông tin trên điện thoại di động là cần thiết.

Song song với sự phát triển của điện thoại di động, ngày càng có nhiều hệ điều hành phát triển trên điện thoại di động. Ngoài các hệ điều hành phổ biến như Symbian, Window Mobile, Palm OS, còn có hệ điều hành dựa trên mã nguồn mở Linux như Limo, Android. Android là một nền tảng mới của Google dành cho thiết bị di động. Mạc dù, đến thời điểm hiện tại (tháng 7.2008) vẫn chưa có thiết bị cụ thể nào chạy hệ điều hành Android. Nhưng Android gây tiếng vang lớn trong cộng đồng phát triển và hứa hẹn một tương lại phát triển trên điện thoại di động.

Xuất phát từ các lý do trên, nhóm em đã thực hiện đề tài "**Ứng dụng bảo mật trên G-Phone",** với mục tiêu chính tìm hiểu nền tảng, kiến trúc, kỹ thuật xây dựng ứng dụng trên môi trường Android và xây dựng thử nghiện ứng dụng bảo mật cho điện thoại di động chạy hệ điều hành Android.

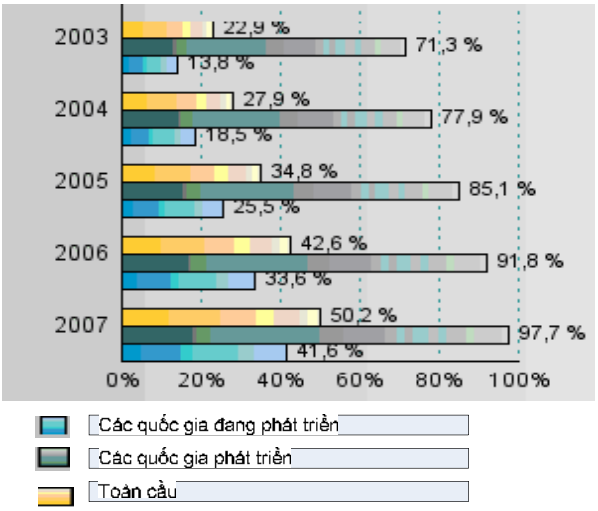
Chương 1

Mở đầu

✍ *Nội dung của chương 1 trình bày tổng quan về nhu cầu bảo mật trên thiết bị di động và một số môi trường phát triển ứng dụng trên thiết bị di động, đồng thời giới thiệu mục tiêu đặt ra của đề tài.*

1.1. Nhu cầu thực tế

Trong cuộc sống, nhu cầu thông tin liên lạc hết sức cần thiết. Có nhiều phương pháp nhằm thực hiện nhu cầu này, từ thư tín, điện thoại bàn đến thư điện tử, điện thoại di động. Trong đó, điện thoại di động nổi bật lên như một phương tiện liên lạc hữu ích nhất, tiện lợi nhất, đặc biệt đối với những người sống và làm việc trong các đô thị. Nhờ chức năng đàm thoại trực tiếp mọi lúc mọi nơi, điện thoại di động ngày càng được sử dụng rộng rãi.



Hình 1.1: Tỉ lệ thâm nhập điện thoại di động trên thế giới (màu vàng). Màu xanh nhạt: tại các quốc gia đang phát triển. Màu xanh đậm: tại các quốc gia phát triển.

Nguồn: Idate.fr, 2/2008

Hiện nay, trên thế giới, điện thoại di động phát triển không ngừng. Hàng loạt điện thoại với các tính năng, ứng dụng hiện đại được tung ra thị trường. Điện thoại di động là thiết bị điện tử phát triển nhanh cả về công nghệ lẫn tính năng, ứng dụng. Riêng ở Việt Nam, thị trường điện thoại di động đã và đang phát triển mạnh mẽ với số lượng người sử dụng đông đảo.

Cùng với việc phát triển nhanh chóng của điện thoại, yêu cầu của người sử dụng với điện thoại di động ngày càng cao. Ngoài các chức năng cơ bản như nghe gọi, gởi và nhận tin nhắn, điện thoại còn hỗ trợ nhiều tính năng khác như nghe nhạc, sổ tay ghi chú… Do đó, dữ liệu lưu trữ trên điện thoại ngày càng gia tăng. Ngoài các dữ liệu thông thường như tin nhắn, thông tin liên lạc, còn có các loại dữ liệu khác như nhạc, hình ảnh, thông tin ghi chú… Và trong nhiều tình huống, người sử dụng lưu trữ nhiều thông tin quan trọng, nhạy cảm trên điện thoại di động như thông tin về công ty, hợp đồng hay đối tác... Lưu trữ thông tin trên điện thoại giúp cho thông tin quản lý dễ dàng ở mọi lúc, mọi nơi. Khả năng này đem lại nhiều tiện lợi cho người sử dụng nhưng đồng thời cũng mang lại nhiều rủi ro cao khi dữ liệu trong các thiết bị di động bị đánh cắp, hay điện thoại bị mất sẽ làm lộ nhiều thông tin quan trọng. ***Do đó, nhu cầu về ứng dụng bảo mật dữ liệu trên thiết bị di động là cần thiết.***

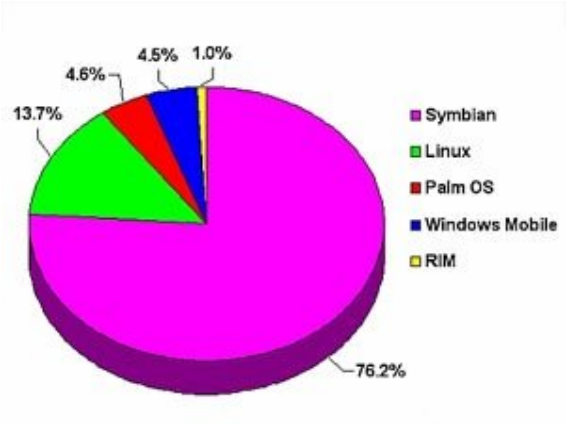
Cùng với sự phát triển của điện thoại, các hệ điều hành trên điện thoại cũng ngày càng phát triển. Ngoài các hệ điều hành phổ biến như Symbian, Window Mobile, Palm, còn có các hiệu điều hành mới như Android, OS X (Iphone)… Phần tiếp theo sẽ giới thiêu sơ lược về các hệ điều hành nói chung và hệ điều hành Android trên thiết bị di động.

1.2. Sơ lược về hệ điều hành trên thiết bị di động

Hiện nay có rất nhiều hệ điều hành trên các thiết bị di động như Symbian, Window Mobile, Palm OS, OS X và hệ điều hành khác dựa trên mã nguồn mở như Android, Limo… Trong đó, Symbian và Window Mobile là hai hệ điều hành phổ biến trên các thiết bị di động.

- **Windown Mobile** được phát triển từ một nhánh hệ điều hành khác của Microsoft. Windows Mobile được phát triển từ các phiên bản Windows CE (từ năm 1998) dùng trên handheld PC .Windows CE sau đó được đổi tên thành Windows Mobile. Các tính năng đặc trưng và phổ biến của Windows Mobile mang phong cách giống như các phiên bản khác của Windows: cũng có giao diện tương tự, có chương trình hỗ trợ văn phòng Office Mobile (bao gồm Word, Exel, Power Point) và có khả năng đồng bộ hóa với máy tính, Outlook cho việc gởi và nhận email, Window Media Player…
- **Symbian** là hệ điều hành độc quyền và phát triển chỉ cho các thiết bị di động.Các thư viện lập trình liên quan, giao diện người dùng hay những công cụ phát triển được cung cấp độc quền bởi công ty Symbian Ltd. Số lượng điện thoại sử dụng Symbian đang chiếm thị phần lớn nhất. Mục đích chính của Symbian OS khi được thiết kế là chuyên dùng cho các thiết bị cầm tay với những nguồn tài nguyên (bộ nhớ, CPU…) bị giới hạn mà có thể chạy hàng tháng hay hàng năm, tối ưu và giảm thiểu các rủi ro trong bộ nhớ… Chính vì vậy, cấu trúc hoạt động của Symbian gây khá nhiều khó khăn cho các lập trình viên khi tìm hiểu về nó.

Ngoài ra còn có hệ điều hành khác như OS X (hệ điều hành của Apple dành cho IPhone), Palm OS và các hệ điều hành dựa trên nhận Linux. Khác với Symbian, hay Windows Mobile yêu cầu chi phí bản quyền, các hệ điều hành dựa trên nhận Linux hiện hoàn toàn miễn phí và Android là một trong số đó.



Hình 1.2: Thị phần các hệ điều hành phổ biến trên điện thoại di động.

(Nguồn:*Gartne.comr)*

Android là nền tảng phần mềm dựa trên mã nguồn mở Linux OS (Kernel 2.6) cho máy di động và những phần mềm trung gian (middleware) để hỗ trợ các ứng dụng mà người sử dụng cần đến. Android được Google công bố lần đầu tiên vào ngày 12 tháng 11 năm 2007, bao gồm:

- Nền tảng hệ điều hành Android
- Công cụ phát triển ứng dụng dựa trên nền tảng Android
- Máy ảo.

Các phiên bản Android tính thời điểm tháng 7 năm 2008:

Phiên bản	Thời điểm phát hành
Phiên bản m3-rc20a	Ngày 12 tháng 11 năm 2007
Phiên bản m3-rc22a	Ngày 16 tháng 11 năm 2007
Phiên bản m3-rc37a	Ngày 14 tháng 12 năm 2007
Phiên bản m5-rc14	Ngày 12 tháng 12 năm 2008
Phiên bản m5-rc15	Ngày 03 tháng 03 năm 2008

Bảng 1.1: Các phiên bản Android

(Nguồn: http://code.google.com/android/download_previous.html)

Hình 1.3: Máy ảo Android

Android được thiết kế sao cho các nhà phát triển có thể tận dụng tối đa lợi thế của thiết bị cầm tay để xây dựng nên các ứng dụng thực sự thuyết phục. Android hoàn toàn là một hệ điều hành mở. Một ứng dụng có thể gọi thực hiện các chức năng lõi trong điện thoại như thực hiện cuộc gọi, gửi tin nhắn SMS, dùng camera…

Android còn cung cấp sẵn các dịch vụ định vị (đây là một thế mạnh của Google) cùng với một tập các ứng dụng về "Map" với các tính năng rất ấn tượng như dò đường, tính khoảng cách địa điểm, tìm bạn… Google còn dự định tích hợp các dịch vụ sẵn có của hãng lên Android, trong đó phải kể đến Google Maps, dịch vụ bản đồ trực tuyến từ lâu đã rất quen thuộc với người dùng, người dùng sẽ thực sự bị thuyết phục bởi các ứng dụng độc đáo này.

Có thể nói rằng Android đã cung cấp cho chúng ta khá đầy đủ các công cụ cơ bản, giúp các nhà phát triển có thể tự xây dựng các ứng dụng cho riêng mình một cách dễ dàng.

1.3. Mục tiêu của đề tài

Hệ điều hành Android là một hệ điều hành mới hiện nay trên dòng thiết bị di động. Việc tìm hiểu và làm việc trên nó hiện thu hút khá nhiều lập trình viên trong nước cũng như trên thế giới. Trong đề tài này, chúng em tìm hiểu về Android và xây dựng một bộ ứng dụng bảo mật thông tin trên môi trường mới này.

Mục tiêu của đề tài:

- Tìm hiểu về hệ điều hành Android và khả năng lập trình trên môi trường này, đi sâu tìm hiểu cách thức lập trình trên điện thoại thông minh Android.
- Tìm hiểu các kỹ thuật xử lý Contact, SMS trên môi trường Android.
- Nghiên cứu, đề xuất giải pháp và quy trình để bảo mật thông tin Contact, SMS trên nền tảng Android.
- Xây dựng bộ ứng dụng bảo mật thông tin người dùng trên điện thoại bao gồm: ứng dụng GPContact (Bảo mật thông tin contact), ứng dụng GPSms (Bảo mật thông tin SMS).

1.4. Nội dung khóa luận

Nội dung luận văn được trình bày trong 09 chương; trong đó 5 chương đầu trình bày các vấn đề về hệ điều hành Android và các kỹ thuật lập trình trên hệ điều hành này. Các chương cuối tập trung vào bộ ứng dụng bảo mật

- **Chương 1. Mở đầu:** trình bày nhu cầu thực tế, lý do thực hiện đề tài và các mục tiêu cần đạt được.
- **Chương 2. Tìm hiểu chung về Android:** giới thiệu về hệ điều hành Android và công cụ phát triển phần mềm. Trình bày kiến trúc một ứng dụng.
- **Chương 3. Lập trình giao diện trong Android:** trình bày một số kỹ thuật liên quan tới giao diện trong Android
- **Chương 4. Các kỹ thuật xử lý:** trình bày một số giải pháp kỹ thuật liên quan tới tin nhắn SMS, cuộc gọi bao gồm truy xuất và bảo mật trên tin nhắn, cuộc gọi.
- **Chương 5. Một số quy trình đề nghị để bảo mật thông tin trên điện thoại di động:** đưa ra một số quy trình bảo mật liên quan tới việc trao đổi khóa, mã hóa nội dung tin nhắn, cuộc gọi mà nhóm đã tìm hiểu
- **Chương 6. Giới thiệu ứng dụng:** giới thiệu về hai ứng dụng GPContact và GPSms mà nhóm đã phát triển dựa trên các quy trình đã xây dựng trong chương 5.
- **Chương 7. Kiến trúc hệ thống và chức năng ứng dụng GPContact:** mô tả kiến trúc của ứng dụng GPContact đã xây dựng theo quy trình đã trình bày ở chương 5 và chức năng đã trình bày ở chương 6.
- **Chương 8. Kiến trúc hệ thống và chức năng ứng dụng GPSms:** mô tả kiến trúc của ứng dụng GPSms đã xây dựng theo quy trình đã trình bày ở chương 5 và chức năng đã trình bày ở chương 6.
- **Chương 9. Tổng kết:** trình bày những kết quả đạt được và hướng phát triển trong tương lai

Chương 2

Tổng quan về Android

✍ *Nội dung của chương 2 giới thiệu tổng quan về hệ điều hành Android:*

- *Giới thiệu về hệ điều hành Android và công cụ phát triển phần mềm*
- *Kiến trúc ứng dụng trên hệ điều hành Android*

2.1. Giới thiệu hệ điều hành Android

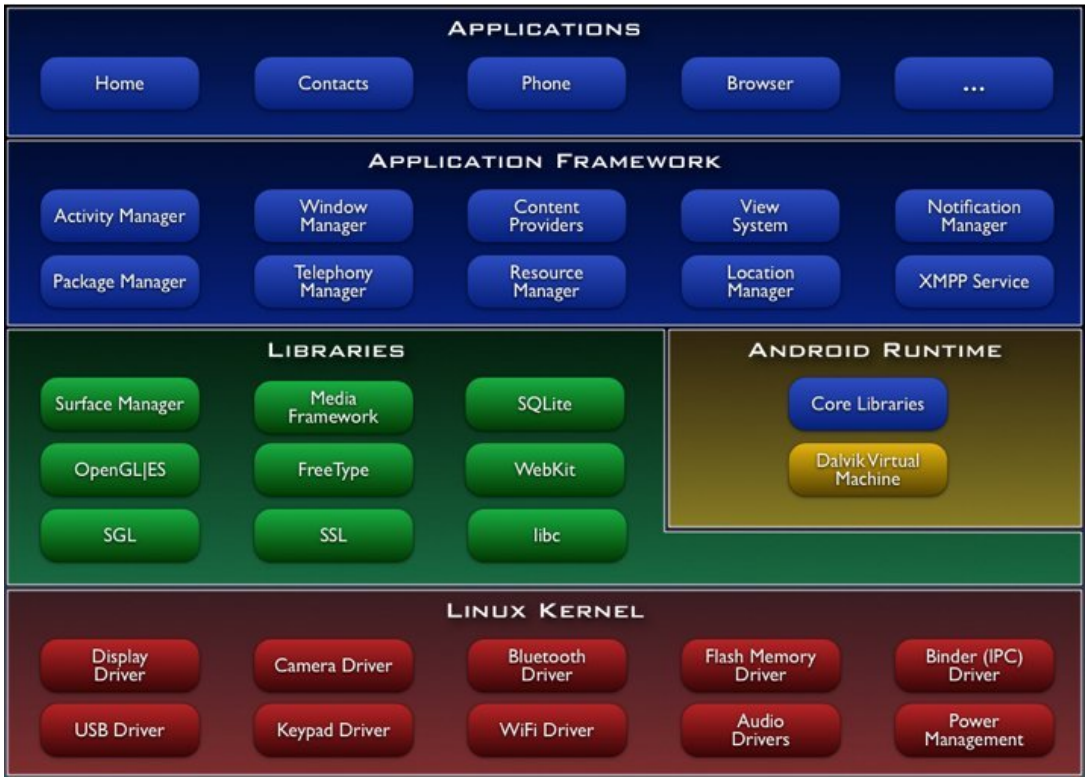
2.1.1. Lịch sử phát triển

Hình 2.1 Các công ty tham gia trong liên minh thiết bị cầm tay mở

Từ cuối năm 2004, trong giới công nghệ thông tin và truyền thông đã bàn về một loại máy điện thoại di động "GPhone" hay "Google phone" có nhiều chức năng nổi bật. Lời tuyên bố ngày 05/11/2007 bởi chính vị lãnh đạo số 1 của Google - ông

Eric Schmidt, khi giới thiệu về công nghệ Android đã xóa tan những nghi vấn trong thế giới công nghệ thông tin và truyền thông. Khác với tập đoàn Apple đã đưa vào thị trường toàn cầu một sản phẩm độc đáo có tên là "iPhone", Google mạnh hơn nữa, đã liên kết 33 tập đoàn và công ty lớn danh tiếng trên thế giới dưới danh hiệu "Open Handset Alliance" (OHA) để tạo một thế đứng mạnh mẽ và vững chắc, rồi cùng nhau sáng tạo và phát triển Công Nghệ Phần Mềm "Android" cho các loại máy di động tương lai trong một thị trường rất rộng lớn (3 tỷ máy) để cạnh tranh với những đối thủ đáng nể như Symbian, Microsoft, Palm...

2.1.2. Tổng quan kiến trúc Android



Hình 2.2 Kiến trúc hệ điều hành Android

[nguồn: android-sdk_m5-rc14/docs/what-is-android.html]

Hệ điều hành Android được chia thành nhiều tầng, mỗi tầng bao gồm một tập các thư viện hoặc các ứng dụng tương ứng.

• **Ứng dụng cơ bản:** Android sẽ có các ứng dụng chính như email, chương trình quản lý tin nhắn SMS, lịch, bảng đồ, trình duyệt, Contact. Tất cả các ứng dụng được viết dựa trên ngôn ngữ lập trình java.

• **Khung ứng dụng:** Nhà phát triển có khả năng can thiệp sâu vào các API (các API được dùng trong các ứng dụng lõi). Kiến trúc ứng dụng được thiết kế để đơn giản hóa việc tái sử dụng các thành phần; bất kì ứng dụng nào cũng có thể đưa ra các chức năng của mình một cách công khai và các ứng dụng khác sau đó có thể sử dụng các chức năng này (vấn đề bảo mật vẫn được tôn trọng). Cơ chế này cho phép người dùng hiệu chỉnh được các thành phần.

Bên dưới tất cả các ứng dụng là một tập các dịch vụ hệ thống, bao gồm:

- Một tập các khung nhìn (View) có thể được sử dụng để xây dựng ứng dụng như list, grid, text box, button, và trình duyệt web nhúng.
- Content Provider cho phép các ứng dụng có khả năng truy cập dữ liệu từ các ứng dụng khác (ví dụ như contact), hay chia sẽ dữ liệu của chính bản thân ứng dụng.
- Trình quản lí tài nguyên, cung cấp khả năng truy cập tới các tài nguyên phi – code như chuỗi, đối tượng đồ họa, và các file layout
- Notification Manager: cho phép các ứng dụng có khả năng thể hiện các thông báo trong thanh trạng thái
- Activity Manager: quản lí chu kì sống của ứng dụng

• **Máy ảo Dalvik:** Máy ảo của Android có tên là Dalvik, chiếc máy ảo dựa trên việc đăng kí (register-based) và được thiết kế bởi Dan Bornstein và một số kĩ sư của Google. Thuật ngữ "register-based" đã nói lên sự khác biệt giữa máy ảo Android với máy ảo java thông thường. Tương tự như máy ảo java, đây cũng là một máy ảo thông dịch, nhưng thông dịch các file có định dạng **.dex** (Dalvik Execute) – một định dạng giúp tối ưu bộ nhớ và khả năng lưu trữ của máy ảo – Các máy ảo java ngày nay chủ yếu dựa trên ngăn xếp (Stack-based), còn máy ảo Android thì khác dựa trên việc đăng kí (Register-based), với tính chất này cho phép

rút ngắn thời gian thực thi chương trình (Các chương trình có thể 'phình' ra sau khi được biên dịch). Tuy nhiên, trong các phiên bản Android hiện nay thì máy ảo vẫn chưa hoàn thiện, nó chưa hỗ trợ kết nối bluetooth, USB, không hỗ trợ việc gọi và nhận cuộc gọi thực sự mà chỉ thông qua trình giả lập (console hoặc IDE hỗ trợ). Có thể trong các phiên bản sau Android sẽ bổ sung các tính năng nhằm hoàn thiện hơn Android

- **Thư viện:** Android bao gồm một tập thư viện C/C++ được sử dụng bởi nhiều thành phần của hệ thống Android. Một vài thư viện chính sẽ được trình bày dưới đây
    - **System C library** – một thư viện bắt nguồn từ BSD của thư viện chuẩn C (libc), giành cho các thiết bị nền Linux
    - **Media Libraries** – dựa trên PacketVideo's OpenCore; những thư viện hỗ trợ playback, có các định dạng video và audio thông dụng, cũng như các file ảnh tĩnh bao gồm : MPEG4, H.264, MP3, AAC, AMR, JPG , và PNG
    - **Surface Manager** – Quản lí việc tương tác với các màn hình 3D, 2D
    - **LibWebCore** – Một thư viện gia tăng sức mạnh cho cả hai loại trình duyệt bao gồm trình duyệt Android và trình duyệt dạng nhúng (webview).
    - **SGL** – Thư viện đồ họa 2D .
    - **3D libraries** – thư viện dựa trên tập APIs của OpenGL ES 1.0.
    - **Free Type** - bitmap và vector font rendering.
    - **SQLite** – hệ quản trị cơ sở dữ liệu quan hệ gọn nhẹ và mạnh mẽ.
- **Android RunTime:** Android bao gồm một tập các thư viện lõi cung cấp hầu hết các chức năng có sẵn trong tập thư viện lõi của ngôn ngữ lập trình java
- **Linux Kernel:** Android dựa trên Linux phiên bản 2.6 cho các dịch vụ lõi như bảo mật, quản lí bộ nhớ , quản lí tiến trình, ngăn xếp mạng, và các driver model. The kernel cũng đóng vai trò như một lớp trừu tượng giữa phần cứng và các phần mềm còn lại .

2.1.3. Các thiết bị sử dụng và tương lai của Android

Hiện nay, việc một chiếc điện thoại chạy nền Android đang được các công ty khác sản xuất như HTC, Samsung và LG.

Hình 2.3 Mẫu điện thoại thông minh Android

Tính tới thời điểm này hãng đang hợp tác với 33 công ty, trong đó có những hãng sản xuất di động như Motorola, Samsung Electronics và High Tech Computer.

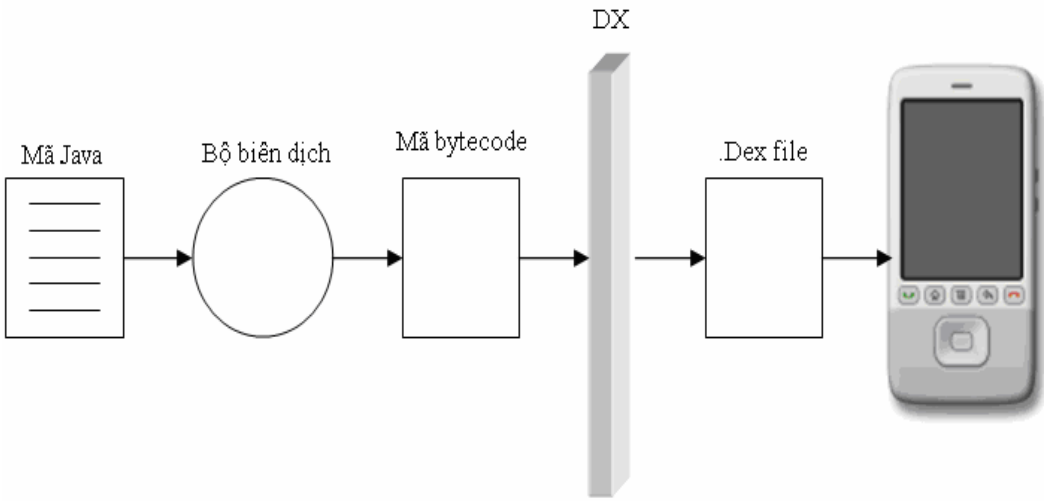
"Chúng tôi hy vọng sẽ có tới hàng ngàn chiếc điện thoại di động khác nhau sử dụng Android" – Giám đốc điều hành Google Eric Schmidt phát biểu trong cuộc họp báo sau khi đưa ra thông báo. Động thái này của Google đã đặt hãng vào vị trí cạnh tranh với Nokia, Microsoft và một trong những đối tác của Google là Apple. Hãng nghiên cứu Strategy Analytics dự đoán Android sẽ chiếm khoảng 2% thị phần thị trường smartphone trong năm 2008. Cũng theo Yankee Group, trong năm nay, smartphone cũng chỉ có được 6% thị phần thị trường điện thoại di động Mỹ.

2.1.4. Ngôn ngữ và công cụ lập trình

Dựa vào các thông tin trên nền tảng Android, chúng ta thấy rằng chỉ có thể phát triển ứng dụng trên điện thoại di động tương lai bằng ngôn ngữ Java (sử dụng thư viện hàm API do hệ điều hành Android cung cấp)

- **Java**

Lập trình bằng ngôn ngữ Java vẫn là một ưu tiên hàng đầu của Google, bởi vì một số ưu điểm của nó, là một ngôn ngữ lập trình đơn giản, được tinh gọn từ C, nên chắc chắn java đơn giản hơn C rất nhiều, java được thiết kế dựa trên Eiffel, Smalltalk, Objective C, Cedar/Mesar là các ngôn ngữ hướng đối tượng mạnh. Điều nó cũng đủ nói lên java là một ngôn ngữ hướng đối tượng. Một ưu điểm khác của java đó là hỗ trợ lập trình phân tán, giúp các lập trình viên có thể truy xuất các máy ở xa thông qua gói java.net. Ngoài ra java còn một loạt ưu điểm như tính thông dịch, mạnh mẽ, bảo mật, kiến trúc trung tính, khả chuyển, hiệu quả cao, đa tuyến, và linh động. Làm việc với java cho phép lập trình viên chỉ cần phát triển ứng dụng mà không quan tâm tới thiết bị cụ thể.



Hình 2.4 Mô hình hoạt động của Android

Trong nền hệ thống Android, máy ảo chạy các file có định dạng .dex, nên mã bytecode thuần java chỉ là một file trung gian, Android cần một công cụ có tên là dx nhiệm vụ chính là chuyển đổi sang các file có định dạng .dex để có khả năng thực thi trên máy ảo.

- **Eclipse**

Hiện nay có vài công cụ, môi trường phát triển (IDE) để xây dựng ứng dụng cho điện thoại di động Android, trong đó có một IDE nổi tiếng và là sự lựa chọn hàng đầu đó là Eclipse với các plugin của Android.

Với IDE này, người phát triển ứng dụng sẽ có một cái nhìn đồ họa trực quan, không phải sử dụng command-line để thao tác. Giúp người phát triển có thể tạo một ứng dụng Android nhanh nhất có thể

2.2. Kiến trúc chung trong ứng dụng Android

2.2.1. Các thành phần trong ứng dụng

Các thành phần phục vụ việc xây dựng một ứng dụng Android bao gồm: **Activity**, **Intent Receiver**, **Service**, **Content Provider**. Không phải ứng dụng nào cũng cần hết 4 thành phần đó, nhưng các ứng dụng sẽ được viết dựa trên sự kết hợp của 4 thành phần này.

Khi quyết định thành phần nào ta cần trong ứng dụng, ta nên liệt kê chúng vào trong một file cấu hình gọi là **AndroidManifest.xml**. Đây là file XML nơi ta khai báo các thành phần của ứng dụng, khả năng và yêu cầu của ứng dụng

- **Activity:**

Đây là một trong các thành phần thông dụng nhất. Activity thường là một màn hình đơn trong ứng dụng. Mỗi Activity sẽ được "kế thừa" từ một lớp cơ sở. Lớp này sẽ thể hiện giao diện người dùng bao gồm các khung nhìn (View) cũng như phải 'hồi đáp' lại tất cả các sự kiện. Hầu như ứng dụng nào cũng có nhiều màn hình. Ví dụ, một ứng dụng gửi tin nhắn phải có màn hình liệt kê danh sách các contact, màn hình thứ 2 để viết tin, và các màn hình khác để xem lại các tin đã gởi hay để thay đổi cấu hình. Mỗi màn hình lúc này sẽ là một Activity. Trong một vài trường hợp, một activity có thể trả lại một giá trị cho màn hình (activity) trước. Ví dụ, một Activity A cho phép người dùng chọn một tấm ảnh rồi chuyển về Activity trước (Activity đã gọi Activity A này)

Khi một màn hình mới được mở, màn hình trước đó sẽ dừng lại và được đặt vào trong history stack. Người dùng hoàn toàn có thể di chuyển ngược lại các màn hình đã mở trước đó. Những màn hình có thể bị chọn để hủy trong history nếu chúng vô ích. Android ghi nhớ history stack cho mỗi ứng dụng được chạy từ màn hình chính

- **Intent và Intent Filter**

Android sử dụng một lớp đặc biệt gọi là Intent để di chuyển từ màn hình này sang màn hình khác. Một intent mô tả một ứng dụng muốn làm gì. Hai thành phần quan trọng trong cấu trúc dữ liệu của Intent đó là "hành động" và "dữ liệu" mà "hành động" đó sử dụng. Giá trị điển hình cho "hành động" đó là MAIN (cửa trước của Activity), VIEW, PICK, và EDIT… "Dữ liệu" được thể hiện theo dạng URI. Ví dụ: muốn xem thông tin contact của một người. bạn phải tạo một intent với "hành động" là VIEW và "dữ liệu" URI đại diện cho người đó.

Có một lớp thường kèm theo với Intent gọi là IntentFilter. Trong khi một Intent dùng để làm một việc gì đó, thì một intentFilter sẽ là các mô tả những Intent nào mà một Activity hiện đang có. Một Activity thể hiện thông tin contact của một người sẽ đưa ra một intentFilter mô tả cách xử lý "hành động" View khi hành động này thực sự xảy ra. Activity sẽ đưa Intent Filter của nó vào tập tin AndroidManifest.xml

Việc di chuyển từ màn hình này sang màn hình khác được thực hiện bằng cách sắp xếp lại các Intent. Để di chuyển tới, một activity gọi startActivity(myIntent). Sau đó hệ thống sẽ xem xét tất cả các intent filter của tất cả các ứng dụng đã được cài đặt trên thiết bị, và một khi đã tìm ra intent filter mô tả phù hợp nhất với tham số myIntent, lúc này hệ thống thực sự sẽ start một Activity lên. Một Activity mới sẽ được cấp một Intent, nhờ quá trình này mà Activity được 'phóng' lên. Quá trình này xảy ra vào thời gian khi startActivity được gọi, điều này có 2 lợi ích:

- Các Activity có thể tái sử dụng các chức năng từ các thành phần khác chỉ đơn giản là yêu cầu theo dạng Intent
- Các Activity có thể được thay thế ở bất kì thời điểm nào bằng một Activity mới với một IntentFilter tương đương

- **Intent Receiver**

Có thể dùng Intent Receiver khi muốn ứng dụng của ta phản ứng lại với các sự kiện từ bên ngoài, ví dụ, khi điện thoại đổ chuông hay khi mạng sẵn sàng, hay tới nữa đêm. IntentReceiver không có giao diện trực quan, mặc dù chúng có thể sử dụng NotificationManager để thông báo người dùng nếu có điều gì thú vị xảy ra. Intent receiver được đăng kí trong tập tin AndroidManifest.xml, nhưng vẫn có thể đăng kí trong code bằng cách dùng Context.registerReceiver(). Ứng dụng của bạn không phải chạy liên tục Intent Receiver. Hệ thống sẽ khởi động ứng dụng nếu cần thiết, khi một Intent Receiver bị kích hoạt. Những ứng dụng có thể truyền Intent của chúng tới các ứng dụng khác với Context.broadcastIntent().

- **Service**

Là một đoạn code thực thi không có giao diện. Ví dụ tốt nhất cho điều này là việc chơi một bản nhạc từ một danh sách. Trong một ứng dụng nghe nhạc, sẽ có một hay nhiều activity cho phép người dùng chọn các bài hát và chơi chúng. Tuy nhiên, việc chơi một bản nhạc không nên để một activity xử lý bởi vì người dùng mong muốn bản nhạc vẫn tiếp tục được chơi khi họ qua một màn hình khác. Trong trường hợp này, activity phải khởi động một dịch vụ **Context.startService()** để chạy nền bên dưới nhằm duy trì bài nhạc. Dịch vụ này sẽ chạy cho tới khi kết thúc. Chú ý người dùng có thể kết nối các dịch vụ (hãy khởi động nếu dịch vụ sau khi kết nối không chạy) bằng phương thức **Context.bindService()**. Sau khi kết nối một dịch vụ, người dùng có thể giao tiếp với nó thông qua một giao tiếp interface do chính service đưa ra.

- **Content Provider**

Các ứng dụng có thể lưu trữ dữ liệu trong file, trong cơ sở dữ liệu SQLite, hay bất kì một cơ chế nào khác. Tuy nhiên, một content provider là một lớp cài đặt (implement) một tập các phương thức chuẩn cho phép các ứng dụng khác lưu trữ và truy vấn loại dữ liệu do Content Provider xử lý.

2.2.2. Các tập tin tài nguyên

Android chia tập tin tài nguyên ra làm hai dạng chính: dạng xml và dạng tập tin import từ ngoài và tập trung chúng vào một thư mục chung có tên là res

Trong thư mục res bao gồm:

- Thư mục anim: chứa các tập tin XML định nghĩa các hiệu ứng.
- Thư mục drawable: chứa các tập tin hình ảnh định dạng png, bmp, jpg…
- Thư mục layout: chứa các tập tin XML định nghĩa các giao diện người dùng.
- Thư mục values: chứa các tập tin định nghĩa các giá trị chuỗi, màu sắc…

Thư mục raw: chứa các file phim, file âm thanh có các định dạng thông dụng như wav, mp3…

2.3. Chu kỳ sống của ứng dụng

Thông thường thì các ứng dụng Android chạy trên một tiến trình Linux của riêng ứng dụng. Tiến trình này được tạo khi ứng dụng bắt đầu khởi động, và nó vẫn tiếp tục chạy cho tới khi nào không cần thiết và lúc này hệ thống sẽ thu hồi bộ nhớ và cấp phát cho ứng dụng khác.

Một tính năng quan trọng và đặc biệt của Android là thời gian sống của ứng dụng không được kiểm soát bởi chính ứng dụng đó. Thay vào đó, việc đó sẽ được quyết định dựa vào sự kết hợp giữa hệ thống và các thành phần trong ứng dụng, chính sự kết hợp này sẽ giúp hệ thống biết được tiến trình nào đang chạy, mức độ quan trọng và tổng dung lượng bộ nhớ của có sẵn

Người phát triển ứng dụng phải hiểu được sự khác biệt giữa các thành phần (đặc biệt là Service và IntentReceiver), sự va chạm thời gian sống của các tiến trình ứng dụng. Việc sử dụng các thành phần không hợp lý có thể dẫn tới kết quả hệ thống sẽ "kill" tiến trình ứng dụng trong khi nó đang làm công việc quan trọng.

Một lỗi thường thấy về chu kì sống của hệ thống là.một IntentReceiver sẽ sinh ra một tiểu trình khi nó nhận một Intent trong phương thức onReceiveIntent(), và sau đó thoát ra bằng câu lệnh "return". Chính vì điều này, hệ thống coi như IntentReceiver không còn hoạt động, và vì vậy tiến trình cũng không còn cần thiết.

Vậy thì nó có thể hủy tiến trình bất cứ lúc nào, đồng thời ngắt hoạt động của các tiểu trình từ tiến trình này. Giải pháp cho vấn đề này đó là ta khởi động một service từ IntentReceiver, để hệ thống biết rằng vẫn có một công việc đang hoạt động trong tiến trình

Để quyết định xem tiến trình nào nên bị hủy khi bộ nhớ đầy, Android sẽ dựa vào thứ bậc của tiến trình dựa trên những thành phần đang chạy trên tiến trình và trạng thái của các thành phần đó.

- Tiến trình đang giữ Activity đỉnh (trên cùng trong ngăn xếp) được gọi là **Foreground process**, quản lí màn hình mà người dùng đang tương tác (phương thức onResume() đã được gọi) hay một IntentReceiver đang chạy (phương thức onReceiveIntent() đang thực thi). Chỉ có một vài tiến trình như thế trong hệ thống. Những tiến trình này chỉ sẽ bị hủy trong trường xấu nhất đó là bộ nhớ quá chậm đến nỗi không còn một tiến trình nào có thể tiếp tục. Nói chung, ở thời điểm này thì hệ thống đạt đến một trạng thái gọi là phân trang bộ nhớ, vì vậy hành động này phải được thực hiện để có thể giữ tính "hồi đáp" của giao diện người dùng.
- Tiến trình giữ activity hiển thị trên màn hình nhưng không phải là Foreground (phương thức onPause() của nó được gọi) được gọi là **Visible Process**. Ví dụ, một foreground activity đã hiển thị một hộp thoại, lúc này ta vẫn còn có thể thấy các activity phía sau hộp thoại nhưng ta hoàn toàn không tương tác được. Những tiến trình như thế sẽ được coi là quan trọng và sẽ không bị hủy nhưng thứ bậc của nó vẫn thấp hơn các tiến trình kiểu foreground.
- Tiến trình giữ dịch vụ (service) đã được khởi động bởi phương thức startService() được gọi là **Service process**. Mặc dù các tiến trình này không trực tiếp hiển thị cho người dùng, nhưng nó cũng làm các công việc quan trọng. Vì vậy mà hệ thống sẽ luôn giữ các tiến trình như thế trong trường hợp còn đủ bộ nhớ để lưu lại các visible và foreground activity.
- Tiến trình giữ các Activity không hiển thị cho người dùng (phương thức onStop() của có được gọi) được gọi là **Background process**. Theo kinh nghiệm thì các tiến trình này không tương tác trực tiếp với người dùng. Hệ

thống sẽ hủy các tiến trình này bất cứ lúc nào cho 3 loại tiến trình trước cần sử dụng bộ nhớ. Thường thì có nhiều tiến trình loại này, vì vậy chúng ta dùng một danh sách LRU để đảm bảo tiến trình nào được sử dụng gần nhất bởi người dùng chính là tiến trình sau cùng bị hủy khi bộ nhớ không đáp ứng đủ

- Tiến trình không giữ bất cứ thành phần ứng dụng gì được gọi là tiến trình rỗng (**empty process**). Lí do để giữ các tiến trình như thế là vì nó đóng vai trò bộ nhớ đệm nhằm cải tiến thời gian startup cho lần kế tiếp khi một thành phần của ứng dụng cần. Hệ thống sẽ thường hủy các tiến trình này để cân bằng tài nguyên hệ thống giữa các tiến trình đệm rỗng này với các bộ đệm nhân bên dưới

Khi quyết định phân loại một tiến trình, hệ thống sẽ thu lượm các cấp độ quan trọng của tất cả các thành phần đang "active" trong tiến trình. Cần tham khảo thêm tài liệu về Activity Service, và IntentReceiver để biết thêm chi tiết về việc mỗi thành phần đóng góp như thế nào trong toàn bộ chu kì sống của tiến trình

Độ ưu tiên của tiến trình có thể gia tăng dựa trên các phụ thuộc của nó đối với tiến trình. Ví dụ, nếu một tiến trình A đã nối kết tới một dịch vụ (Service) với cờ **Context.BIND_AUTO_CREATE** hoặc đang sử dụng Content Provider trong tiến trình B, thì tiến trình B sẽ kém quan trọng hơn tiến trình A.

2.4. Kết luận

Trong chương này ta đã trình bày sơ lược về hệ điều hành Android, tương lai của hệ điều hành và một số công cụ hỗ trợ. Chúng em nhận thấy rằng trong điện thoại di động sử dụng hệ điều hành Android rồi sẽ chiếm một thị phần lớn và có khả năng phát triển. Ngoài ra chúng em còn tìm hiểu sơ qua kiến trúc một ứng dụng trên Android của hệ điều hành này. Hiểu những kiến trúc này sẽ giúp chúng ta phát triển ứng dụng một cách nhanh chóng và dễ dàng hơn.

Tuy nhiên, muốn để phát triển ứng dụng thực sự, người lập trình cần biết về giao diện người dùng. Đây là một thành phần căn bản sẽ được trình bày trong chương 3.

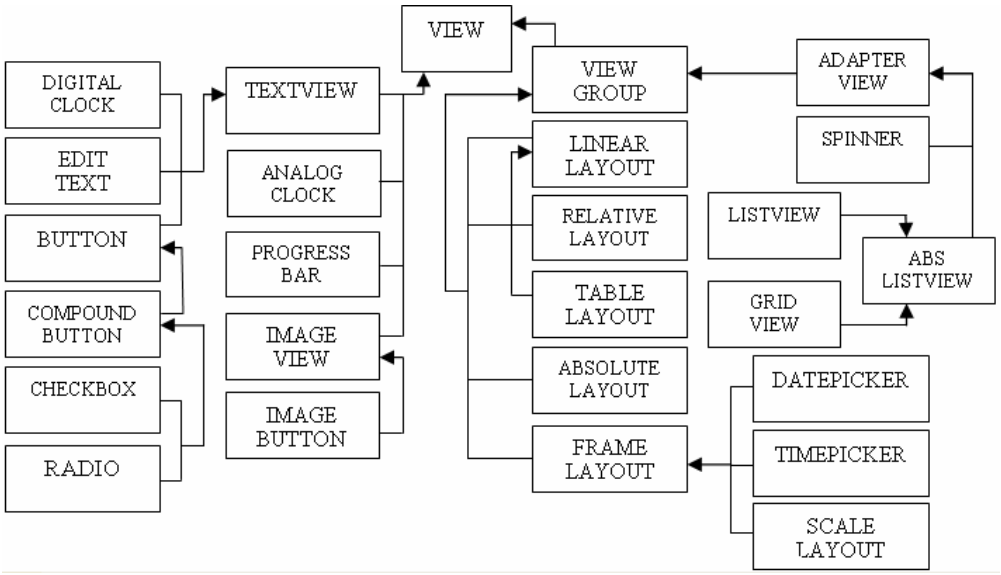
Chương 3

Một số vấn đề về lập trình giao diện trong Android

✍ *Nội dung của chương 3 giới thiệu về mô hình giao diện trên hệ điều hành Android, đồng thời tập trung trình bày một số kĩ thuật và khái niệm cơ bản. Nội dung cụ thể gồm:*

- *Sơ đồ các thành phần giao diện*
- *Các vấn đề cơ bản*
  - *Giới thiệu các layout*
  - *Kết nối dữ liệu*
  - *Cơ chế lắng nghe*
  - *Cách hook các thành phần*

3.1. Giới thiệu



Hình 3.1 Sơ đồ các thành phần giao diện

Java có một bộ thư viện giao diện nổi tiếng Swing, tập hợp các control thông dụng, điều quan trọng là mô hình MVC đã được ứng dụng trong bộ thư viện này. Android đã chọn java là ngôn ngữ lập trình thì không thể bỏ qua mô hình này cho các thành phần giao diện của mình. Android hầu như đặt các thành phần giao diện của mình nằm trong gói Widget và phát triển chúng dựa trên mô hình MVC.

3.2. Hệ thống phân cấp các thành phần

Đơn vị chức năng cơ bản của một ứng dụng Android là activity – một đối tượng của lớp **`android.app.Activity`**. Một activity có thể đảm trách nhiều công việc, nhưng bản thân nó không có một dáng vẻ thật sự. Để có một activity, người dùng cần làm việc với **view** và **viewgroup** – đây là các đơn vị cơ bản của giao diện người dùng trên nền Android.

3.2.1. Khung nhìn

Khung nhìn (View) là một đối tượng của lớp `android.view.View`. Cấu trúc dữ liệu của nó lưu trữ bố cục và nội dung của màn hình. Một View đảm trách hầu hết các công việc cơ bản trên các bố cục (layout) như đo lường, xử lý focus, hay cuộn màn hình…

Lớp khung nhìn là một lớp cơ sở cho widget – đây là một tập các lớp con (subclass) dùng để vẽ các thành phần có khả năng tương tác với màn hình (như button hay textbox). Widget tự xử lý việc vẽ cũng như đo lường, vì vậy ta có thể xây dựng giao diện người dùng một cách nhanh chóng. Danh sách widget bao gồm **Text, EditText, InputMethod, MovementMethod, Button, RadioButton, CheckBox,** và **ScrollView.**

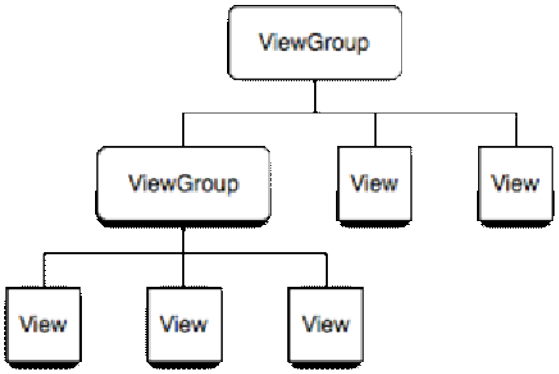
3.2.2. Nhóm khung nhìn

Nhóm khung nhìn (View Group) là một đối tượng của lớp `android.view.Viewgroup`. Đây là một kiểu đối tượng view đặc biệt với chức năng là quản lí một tập con các khung nhìn và nhóm khung nhìn khác. Nhóm khung nhìn cho phép ta thêm vào cấu trúc giao diện màn hình và xây dựng nên các màn hình phức tạp.

Nhóm khung nhìn là một lớp cơ sở cho các bố cục (layout) – đây là một tập các lớp con (subclass) cung cấp các bố cục thông dụng. Các bố cục sẽ giúp ta xây dựng cấu trúc cho một tập các khung nhìn bên trong.

3.2.3. Cấu trúc dạng cây của giao diện người dùng

Để xây dựng nên giao diện người dùng cho một activity, ta có thể dùng các node view và viewgroup như biểu đồ bên dưới. Cây có thể đơn giản hay phức tạp, ta cũng có thể dùng tập widget đã được định nghĩa sẵn hoặc các view do ta tự định nghĩa



Hình 3.2 Cấu trúc cây giao diện

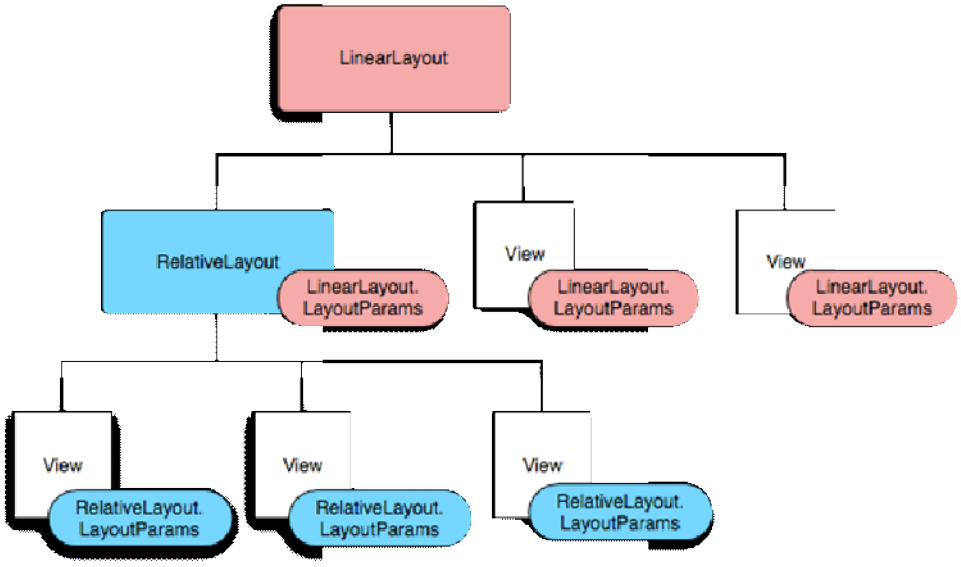
[nguồn: android-sdk_m5-rc14_windows/docs/devel/ui/hierarchy.html]

Để vẽ giao diện ra màn hình trong một activity, ta gọi phương thức **setContentView()** và truyền cho nó tham chiếu tới đối tượng node gốc. Khi hệ thống Android đã có tham chiếu tới đối tượng node gốc, nó có thể làm việc trực tiếp với node để bắt đầu thực hiện việc vẽ toàn bộ cây ra màn hình. Khi activity của ta "active" và nhận được focus, hệ thống sẽ thông báo cho activity và yêu cầu node gốc vẽ cây. Node gốc sau đó lại yêu cầu các node con trực tiếp vẽ cây, mỗi viewgroup sẽ chịu trách nhiệm vẽ các node con trực tiếp của chúng.

Như đã đề cập, mỗi viewgroup sẽ đảm trách việc đo lường, trình bày các node con, và gọi hàm Draw() để bắt các node con tự vẽ chúng. Những node con này có thể 'xin' một kích thước cũng như vị trí từ 'bố mẹ' chúng, nhưng quyết định sau cùng vẫn thuộc về 'bố mẹ'

3.2.4. Tham số đặc tả vị trí và kích thước

Mỗi nhóm khung nhìn sử dụng một lớp lồng kế thừa từ **`ViewGroup.LayoutParams`**. Lớp con này chứa các kiểu thuộc tính sẽ định nghĩa vị trí và kích thước của các thành phần con.



Hình 3.3 Sơ đồ Layout

[nguồn: android-sdk_m5-rc14_windows/docs/devel/ui/hierarchy.html]

Chú ý rằng mỗi lớp con tham số đặc tả có một cú pháp riêng trong việc thiết lập các giá trị. Mỗi thành phần con phải định nghĩa tham số đặc tả.

Tất cả các nhóm khung nhìn đều có các thuộc tính chiều rộng và chiều cao, đôi khi cũng có cả thuộc tính về biên (margin) và đường viền (border). Ta có thể đặc tả độ rộng và độ cao một cách chính xác, mặc dù đôi khi ta không nên làm việc này. Thông thường thì ta muốn các view tự xác định kích thước theo nội dung, hoặc bắt chúng lấp đầy đối tượng chứa chúng

3.3. Layout

Sau đây là một số nhóm khung nhìn (View Group) thông dụng nhất ta sẽ thấy trong các ứng dụng thông thường.

3.3.1. FrameLayout

Đây là một layout đơn giản nhất nhưng lại ít được sử dụng nhất. Các thành phần con đơn giản được gắn vào layout theo thứ tự từ góc trên trái của màn hình, các thành phần con cũng không thể tự đặc tả vị trí của mình, tất cả do layout quyết định. Nếu chúng ta thêm vào layout 2 thành phần trở lên, thì thành phần sau sẽ chồng lên thành phần trước. Tóm lại với layout này ta chỉ được thêm vào một thành phần con duy nhất mà thôi.

Các bước thực hiện tạo layout :

- Bước 1: Thêm vào file xml (“file giao diện”) dòng code sau

```
<?xml version=“1.0” encoding= “utf-8”?>
<FrameLayout xmlns:android=“http://schemas.android.com/apk/res/android”
      android:orientation= “vertical”
      android:layout_width= “fill_parent”
      android:layout_width= “fill_parent”>
      <!—Chỗ thêm thành phần con-->
</FrameLayout>
```

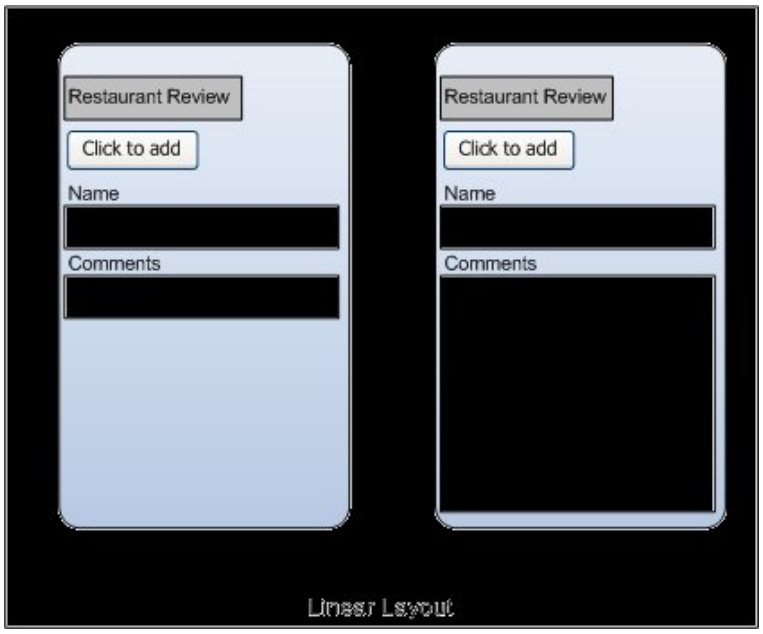
- Bước 2: Sau đó ta thêm các thành phần con vào khoảng trống. Ví dụ là một textView hay button… Như vậy ta đã có một framelayout đơn giản
- Các thuộc tính và phương thức thông dụng.
    - Android:foreground : Giá trị là một tài nguyên kiểu drawable, nếu thiết lập thuộc tính này thì hệ thống sẽ vẽ lên trên hết tất cả các view con trong layout.
    - setForeground(Drawable) : Phương thức này tương tự như trên nhưng dùng trong lúc code chương trình (không dùng trong file xml)

3.3.2. LinearLayout

LinearLayout canh lề tất cả các con theo một hướng nhất định – vertically hoặc horizontally, tùy thuộc vào giá trị của thuộc tính trong layout. Ngoài ra linearlayout cũng tôn trọng các biên giữa các thành phần con, cũng như gravity.

LinearLayout có thuộc tính đặc biệt là weight, với thuộc tính này cho phép các thành phần con 'giãn nỡ' để che lấp đi các khoảng trống vô nghĩa. Ví dụ có 3 textbox, và 2 trong số chúng có weight là 1, thì 2 trong số 3 textbox này sẽ giãn ra để lấp các khoảng trống thừa, còn textbox thứ 3 thì sẽ không phát triển thêm nữa (đúng với kích thước khai báo)

Hai form sau đây đại diện cho một LinearLayout với một tập các thành phần: một button, một vài label, một vài textbox. Tất cả đã có giá trị padding được điều chỉnh một cách hợp lí. Các textbox có thuộc tính width là FILL_PARENT, các thành phần khác là WRAP_CONTENT. Gravity được mặc định là Left. Form bên trái không thiết lập các giá trị weight (mặc định là 0). Form bên phải thì có textbox 'comment' có giá trị được gán là 1, textbox 'NAME' và 'COMMENT' có cùng chiều cao.



Hình 3.4 Hình ảnh về LinearLayout

[nguồn: android-sdk_m5-rc14_windows/docs/devel/ui/layout.html]

Trong một LinearLayout theo chiều ngang, các item sẽ được canh theo vị trí của dòng base line (dòng baseline là dòng đầu tiên của thành phần đầu tiên – trên nhất, trái nhất – và được coi như là dòng tham khảo). Điều này giúp ta lướt qua các thành phần trong form mà không cần di chuyển lên xuống để đọc các đoạn text trong các thành phần lân cận. Ta có thể tắt chức năng canh theo baseline bằng cách thiết lập android:baselineAligned = "false" trong tập tin layout xml.

Các bước để tạo Linear Layout:

- Bước 1: Thêm vào file xml ("file giao diện") dòng code sau:

```
<?xml version="1.0" encoding= "utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation= "vertical"
      android:layout_width= "fill_parent"
      android:layout_width= "fill_parent">
      <!—Chỗ thêm thành phần con-->
</LinearLayout>
```

- Bước 2 : Sau đó ta thêm các thành phần con vào khoảng trống. Ví dụ là một textView hay button… Như vậy ta đã có một linearlayout đơn giản.

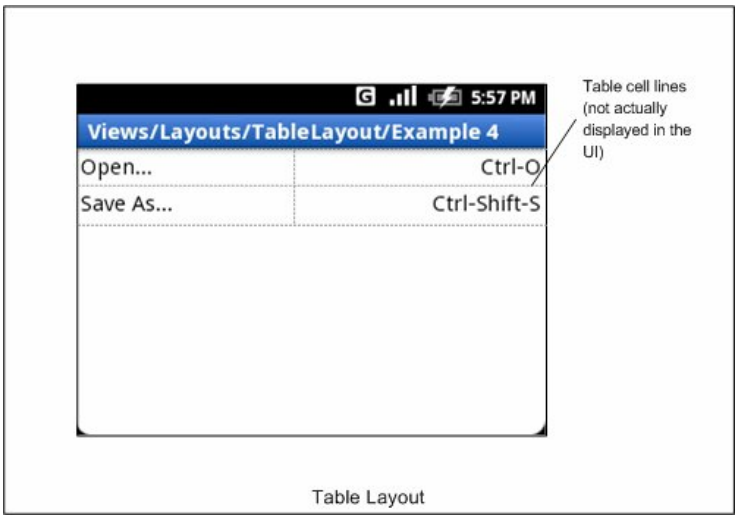
Các thuộc tính và thành phần thông dụng nhất :

- android:gravity: Đặt tả cách đặt một đối tượng theo hai chiều x,y
- setGravity(int): Tương tự như thuộc tính android:gravity
- android:orientation: Nên theo cột hay dòng? Dùng "horizontal" cho dòng, "vertical" cho cột
- setOrientation(int): Tương tự như thuộc tính android:orientation
- android:baselineAligned: Giá trị là true hay false. Khi là false thì chức năng canh theo baseline sẽ bị vô hiệu hóa.
- setBaselineAligned(boolean): Tương tự như thuộc tính android:baselineAligned

3.3.3. Table Layout

Layout này định vị cho các thành phần theo dòng và cột. Một tableLayout bao gồm một tập các đối tượng TableRow, mỗi tablerow định nghĩa một dòng.

TableLayout không thể hiện đường viền. Mỗi dòng có thể không có hoặc có nhiều ô, mỗi ô chứa một đối tượng view. Một bảng có thể để các ô trống



Hình 3.5 Hình ảnh về TableLayout

[nguồn: android-sdk_m5-rc14_windows/docs/devel/ui/layout.html]

Các cột có thể ẩn, hay có thể được đánh dấu giãn ra để lấp đầy các khoảng trống thừa, hay có thể được đánh dấu co lại để bảng vừa với màn hình

Các bước thực hiện tạo layout

- Bước 1: Cấu hình trong file XML.

```
<TableLayout xmlns:android= http://schemas.android.com/apk/res/android
      id= "@+id/layout"
      android:layout_width= "fill_parent"
      android:layout_height= "fill_parent"
      android:stretchColumns= "1">
      <TableRow>
            <-- Các thành phần-->
      </TableRow>
      <TableRow>
            <-- Các thành phần-->
      </TableRow>
</TableLayout>
```

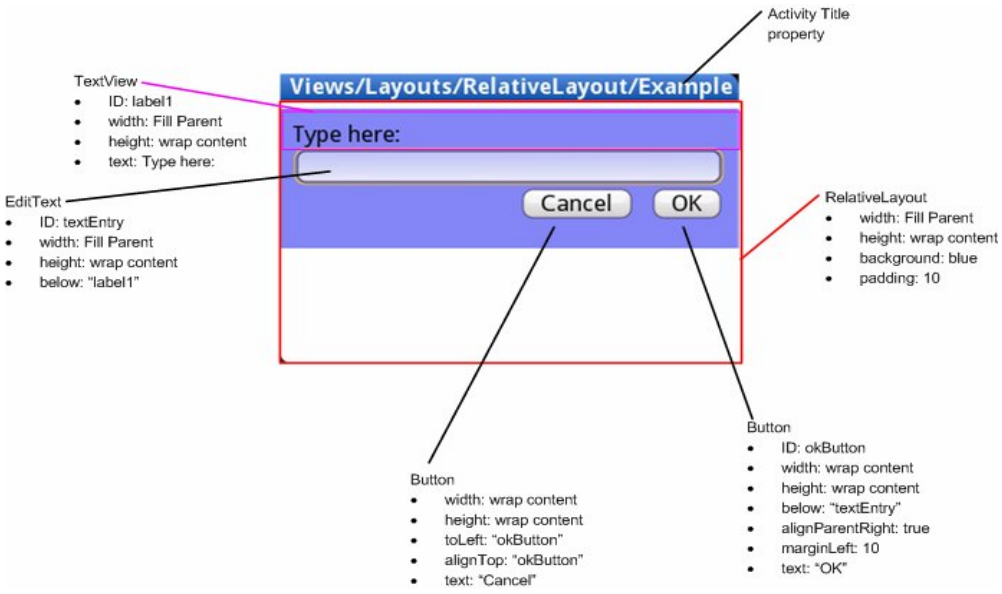
- Bước 2: Sau đó ta thêm các thành phần con vào khoảng trống. Ví dụ là một textView hay button… Như vậy ta đã có một tableLayout đơn giản

Các thuộc tính và thành phần thông dụng:

- android:collapseColumns
- setColumnsCollapsed(int,boolean)
- android:shrinkColumns
- setColumnShrinkable(int,boolean)
- android:stretchColumns
- setColumnStretchable(int,boolean)

3.3.4. Relative Layout

Để cho các thành phần con đặc tả vị trí của chúng một cách tương đối đối với thành phần anh em, hoặc thậm chí đối với đối tượng 'bố mẹ' . Lưu ý một điều quan trọng, phần tử được tham chiếu phải được liệt kê trước khi ta tham chiếu tới nó



Hình 3.6 Hình ảnh về RelativeLayout

[nguồn: android-sdk_m5-rc14_windows/docs/devel/ui/layout.html]

Các bước thực hiện tạo Layout :

- Bước 1 : cấu hình trong file XML

```
<RelativeLayout xmlns:android=
'http://schemas.android.com/apk/res/android'
      android :layout_width= 'fill_parent'
      android :layout_height= 'wrap_content'
      android :background= '@drawable/blue' android :padding= '10dip'>
      < !-- Các thành phần -->
</RelativeLayout>
```

- Bước 2 : Sau đó ta thêm các thành phần con vào khoảng trống. Ví dụ là một textView hay button… Như vậy ta đã có một RelativeLayout đơn giản

Các thuộc tính và thành phần thông dụng :

- android:layout_toRight
- android:layout_toLeft
- android:layout_above
- android:layout_below
- android:layout_alignParentTop
- android:layout_alignParentBottom
- android:layout_alignParentRight
- android:layout_alignParentLeft
- android:layout_alignTop
- android:layout_alignParentBottom

Nhìn chung thì RelativeLayout cũng hoàn toàn tương tự như các layout khác, chỉ khác ở các thuộc tính xml. Đây là một layout khá hiệu quả, không phụ thuộc thiết bị cụ thể, nhưng việc sử dụng hơi gây khó khăn vì cú pháp gần giống như việc ta mô tả bằng lời, dễ gây lầm lẫn

3.3.5. Absolute Layout

Cho phép các thành phần con đặc tả vị trí một cách chính xác tọa độ x,y trên màn hình. Canh lề không được hỗ trợ, các thành phần có thể chồng chéo lên nhau (không khuyến khích). Ta không nên dùng layout này trừ trường hợp bất khả kháng, vì layout này khá cứng nhắc, và không làm việc tốt với các thiết bị khác nhau.

3.4. Kết nối dữ liệu (AdapterViews)

Như đã đề cập, một vài nhóm khung nhìn có UI. Thông thường như Gallery (là một widget lựa chọn hình ảnh) và một ListView (một danh sách các views). Các đối tượng này thường có 2 nhiệm vụ chính:

- Điền dữ liệu vào Layout: Việc làm này rất thường làm, đó là kết nối một lớp với một Adapter, cái sẽ lấy dữ liệu ở đâu đó, có thể là một danh sách do trong code chương trình khởi tạo hoặc cũng có thể truy vấn từ cơ sở dữ liệu

```
//Lấy một spinner và kết nối nó với một ArrayAdapter
//tham chiếu tới một mảng chuỗi
private String[] fruit= {'apple', 'orange', 'lemons'} ;
Spinner s1= (Spinner)findViewById(R.id.fruitlist) ;
s1.setAdapter(new ArrayAdapter<String>(this, R.layout.spinner_1,
mString)) ;
//Kết nối spinner với dữ liệu
private String[] cols= (android.provider.Contacts.PeopleColumns.NAME) ;
private Cursor cur =
managedQuery(android.provider.Contacts.People.CONTENT_URI, cols, null,
null) ;
s2.setAdapter(new CursorAdapter(cur, this)) ;
```

- Xử lý lựa chọn của người dùng : Thông thường nhất thì điều này được thiết lập bởi **AdapterView.OnItemClickListener** để lắng nghe và bắt sự thay đổi lựa chọn

```
//Tạo một đối tượng xử lý thông điệp
private OnItemClickListener mMessageClickedHandler
        = new OnItemClickListener()
{
   public void onItemClick(AdapterView parent,View v,
                           int position,long id)
   {
      //tạo một messageBox
      showAlert('you've got an event', 'Clicked me !', 'ok', false) ;
   }
} ;
//Bây giờ hook vào đối tượng chúng ta và kết nó với sự kiện
onItemClickListener
mHistoryView= (ListView)findViewById(R.id.history);
mHistoryView.setOnItemClickListener(mMessageClickedHandler) ;
```

3.5. Tạo giao diện bằng cấu hình XML

3.5.1. Tạo giao diện

Bởi vì việc thiết kế màn hình bằng code khá cồng kềnh, Android đã hỗ trợ cú pháp xml để thiết kế màn hình. Android đã định nghĩa một lượng lớn các thành phần, mỗi cái đại diện cho một lớp con. Ta có thể thiết kế màn hình giống như cách ta tạo file HTML, tập các tag lồng nhau, được lưu trong thư mục res/layout của ứng dụng.

Mỗi file xml được tạo ra từ các tag tương ứng với các class GUI trong Android. Các tag này có các thuộc tính tương ứng với các phương thức trong class đó (ví dụ: EditText có thuộc tính text tương ứng với EditText.setText)

Chú ý việc tương ứng qua lại giữa các lớp Android với các tag xml là không đúng 100%. Cũng nên chú ý Android có khuynh hướng vẽ các thành phần theo thứ tự nơi 8mà nó xuất hiện trong file XML. Bởi thế, nếu các thành phần có chồng lên nhau, cái sau cùng có thể sẽ được vẽ lên trên tất cả các cái trước ở cùng một vị trí

Mỗi file XML được biên dịch thành một cây có node gốc là một đối tượng view hay viewgroup, và vì vậy phải chứa một tag gốc đơn:

Các đơn vị hỗ trợ về kích thước :

- px (pixel)
- dip(device independent pixel)
- sp (scaled pixel – tốt nhất cho textSize)
- pt (point)
- in (inch)
- mm (millimeter)

Các bước Load giao diện từ file XML :

- Bước 1: Ta tạo một file xml trong thư mục đã res/layout của ứng dụng (Lưu ý tên file là chữ thường)

```
<?xml version= "1.0" encoding= "utf-8"?>
<RelativeLayout xmlns:android= http://schemas.android.com/apk/res/android
       android:layout_width= "fill_parent"
       android:layout_height= "wrap_content"
       android:background= "@drawable/blue"
       android:padding= "10px">
       <TextView id= "@+id/label" android:layout_width= "fill_parent"
              android:layout_height= "wrap_content" android:text= "Type here"/>
       <EditText id= "@+id/entry" android:layout_width= "fill_parent"
              android:background= "@android:drawable/editbox_background"
              android:layout_below= "@id/label"/>
       <Button id= "@+id/ok" android:layout_width= "wrap_content"
              android:layout_height= "wrap_content"
              android:layout_below= "@id/entry"
              android:layout_alignParentRight= "true"
              android:layout_marginLeft= "10px"
              android:text= "OK"/>
       <Button android:layout_width= "wrap_content"
              android:layout_height= "wrap_content"
              android:layout_toLeft= "@id/ok"
              android:layout_alignTop= "@id/ok"
              android:text= "Cancel"/>
</RelativeLayout>
```

- Bước 2: Tải lên bằng cách gọi phương thức **setContentView()** với tham số truyền vào theo chuẩn R.layout.ten_file_xml

```
protected void onCreate(Bundle savedValues)
{
      //Nên gọi lớp cơ sở trước
      super.onCreate(savedValues);
      //Tải file xml đã được biên dịch (ứng dụng tự biên dịch khi ta run
      //chương trình)
      //file nguồn có path là res/layout/hello_activity.xml
      setContentView(R.layout.hello_activity) ;
}
```

3.5.2. Cách Hook các thành phần

Ta có thể lấy một thành phần trong file xml sang cho màn hình bằng cách gọi **Activity.findViewById**. Sau đó ta có thể dùng biến này khôi phục cũng như thiết lập bất cứ giá trị gì của biến đối tượng đó.

Ví dụ:

```
TextView msgTextView = (TextView)findViewById(R.id.msg);
msgTextView.setText(R.string.push_me);
```

3.6. Cơ chế lắng nghe

Có một vài sự kiện tự động được xử lý bởi Android. Ví dụ như Activity có 2 phương thức là onKeyDown và onKeyUp, còn widget có **onFocusChange(boolean,int)**. Tuy nhiên, có một vài sự kiện quan trọng như button clicks, thì ta phải đăng kí

Ví du, xử lý sự kiện button:

```
Public class SendResult extends Activity
{
    /**
     * Initialization of the Screen after it is first created.  Must at
least
     * call setContentView() to
     * describe what is to be displayed in the screen.
     */
    protected void onCreate(Bundle savedValues)
    {
        ...

        // Listen for button clicks.
        Button button = (Button)findViewById(R.id.corky);
```

```
        button.setOnClickListener(mCorkyListener);
    }

    // Create an anonymous class to act as a button click listener.
    private OnClickListener mCorkyListener = new OnClickListener()
    {
        public void onClick(View v)
        {
            // To send a result, simply call setResult() before your
            // activity is finished.
            setResult(RESULT_OK, "Corky!");
            finish();
        }
    };
```

3.7. Kết luận

Trong chương này chúng em đã trình bày một số tìm hiểu về giao diện người dùng trên hệ điều hành Android bao gồm mô hình và một số vấn đề cơ bản. Với những gì đã tìm hiểu trong chương này cho phép chúng ta xây dựng được các ứng dụng một cách nhanh chóng. Mặt khác, do Android là một hệ điều hành cho điện thoại di động, nên có một số kỹ thuật đặc trưng riêng sẽ được trình bày trong chương kế tiếp

Chương 4

Các kỹ thuật xử lý

✍ *Nội dung của chương 4 trình bày các kỹ thuật sử dụng trong đề tài bao gồm các loại kỹ thuật sau:*

- *Kỹ thuật xử lý Contact (phần 4.1)*
- *Kỹ thuật xử lý Activtity (phần 4.2, 4.3)*
- *Kỹ thuật xử lý cuộc gọi (4.4)*
- *Kỹ thuật xử lý tin nhắn SMS (4.5)*
- *Kỹ thuật lưu trữ khóa (4.6)*

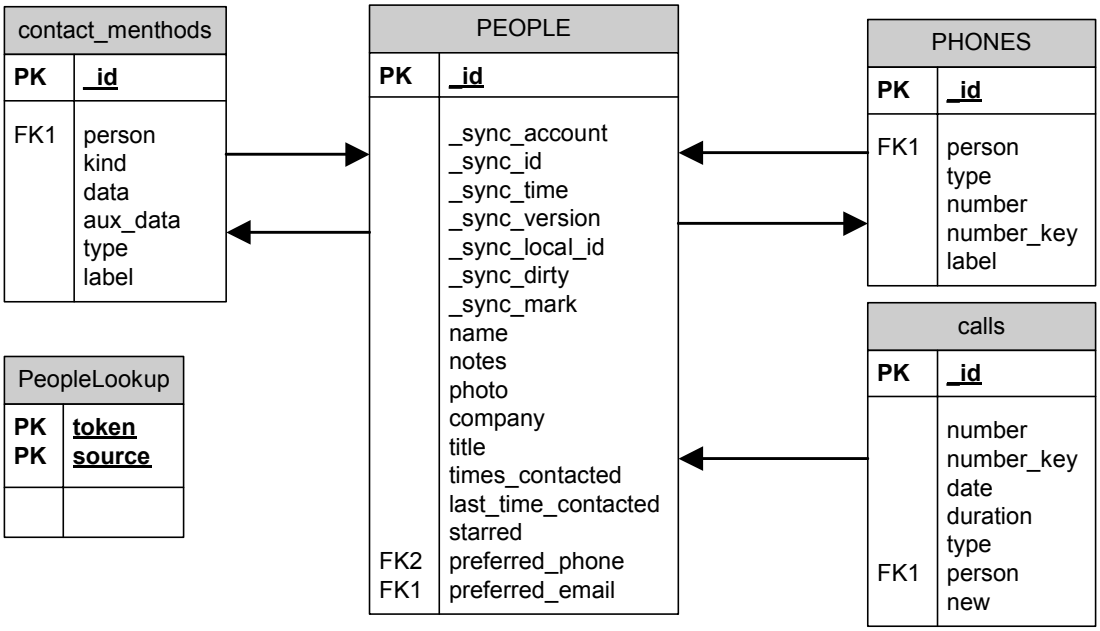
*Nội dung của chương tổng hợp từ tài liệu tham khảo đính kèm với Android SDK phiên bản **m5-rc14** và tổng hợp thông tin từ các ví dụ trong quá trình tìm hiểu.*

4.1. Các kỹ thuật xử lý trong Contact

4.1.1. Mô hình lưu trữ Contact trong Android

Android ***định nghĩa sẵn*** cơ sở dữ liệu lưu trữ Contact và các phương thức chuẩn cho phép các ứng dụng khác có thể truy xuất vào dữ liệu Contact. Android sử dụng cơ sở dữ liệu SQLite để lưu trữ dữ liệu Contact và sử dụng ContentProvider để các ứng dụng khác thông qua nó, có thể truy xuất các thông tin Contact. (Ghi chú là Android không cho phép truy xuất trực tiếp tới cơ sở dữ liệu Contact thông qua các lớp đối tượng xử lý SQLite Database).

Android lưu trữ dữ liệu Contact trong file **Contacts.db (** tập tin **c**ơ sở dữ liệu SQLite)**,** có mô hình Cơ sở dữ liệu như sau :



Hình 4.1: Lược đồ quan hệ của Cơ sở dữ liệu Contact trong Android.

Ngoài các bảng trên, còn có các bảng được thiết kế sẵn cho chức năng đồng bộ hóa (_sync_state, _sync_state_metadata, _delete_people). Dữ liệu Contact trong Android lưu trữ những thông tin tương ứng với các bảng như sau:

- **PEOPLE**

Dữ liệu trong bảng này bao gồm các thông tin sau:

STT	Tên trường	Loại dữ liệu	Ý nghĩa
1	_id	Integer	Id Contact Item.
2	_sync_account	Text	Các trường được Android dành riêng cho việc đồng bộ hóa
3	_sync_id	Text	
4	_sync_time	Text	
5	_sync_version	Text	
6	_sync_local	Integer	
7	_sync_dirty	Integer	
8	_syncmark	Integer	
9	Name	Text	Tên của người dùng.
10	Notes	Text	Thông tin ghi chú về người dùng này.
11	Photo	Text	Hình ảnh của người dùng.
12	Company	Text	Thông tin công ty
13	Title	Text	
14	Times_contacted	Interger	Số lần liên lạc
15	Last_time_contacted	Interger	Thời gian liên lạc lần cuối cùng
16	Started	Interger	
17	Preferred_email	Interger	Địa chỉ email chính, tham chiếu tới bảng Contact_Menthods
18	Preferred_phone	Interger	Số điện thoại chính, tham chiếu tới bảng Phones

Bảng 4.1 : Cấu trúc bảng People trong Cơ sở dữ liệu Contact.

- **PHONES**

Dữ liệu trong bảng này chứa thông tin về số điện thoại của mỗi Contact Item có trong bảng PEOPLE. Bao gồm các thông tin sau:

STT	Tên trường	Loại dữ liệu	Ý nghĩa
1	_id	Integer	Id của số điện thoại.
2	Person	Integer	Thông tin này cho biết số điện thoại này thuộc về người nào trong bảng People. Tham chiếu đến bản PEOPLE.
3	Type	Integer	Loại số điện thoại : 0 – Home 1 – Mobi 2 – Work 3 – Work Fax 4 – Home fax 5 – Pager Với các giá trị trên thì field Label = “”. Trường hợp còn lại để type = 6, lúc đó nhãn label sẽ mang giá trị thích hợp do người dùng nhập vào.
4	Number	Text	Thông tin số điện thoại
5	Number_key	Text	Chưa biết ý nghĩa của trường dữ liệu này, khảo sát nó là số điện thoại bị đảo ngược. Ví dụ : Number : 0913141322 -> Number key : 2231413190
6	label		Với trường hợp type = 6, thì label là nhãn của điện thoại do người dùng nhập vào.

Bảng 4.2: Cấu trúc bảng PHONES trong Cơ sở dữ liệu Contact.

- **CONTACT_METHODS**

Bảng này dùng để lưu các loại dữ liệu khác tương ứng với mỗi Contact Item như Email, Address... Android cho phép các ứng dụng khác có thể lưu trữ dữ liệu khác vào trong bảng này. Bảng này gồm các thông tin như sau:

STT	Tên trường	Loại dữ liệu	Ý nghĩa
1	_id	Integer	Id của thông tin.
2	Person	Integer	Cho biết thông tin này thuộc về người nào nào trong bảng People, tham chiếu tới bảng PEOPLE.

STT	Tên trường	Loại dữ liệu	Ý nghĩa
3	Kind	Integer	Loại dữ liệu, Android quy định như sau Kind = 1 – email Kind = 2 – address
4	Data	Text	Dữ liệu cần lưu
5	Aux_data	Text	Dùng để lưu trữ các dữ liệu khác.
6	Type	Integer	Loai dữ liệu 1. Nếu Kind = 1 (Email) Type = 0 : Home Type = 1: Work Type = 2: Primary Type =3 : Số điện thoại khác 2. Nếu Kind =2 : (Address) Type = 0 : Postal Type =1 : Home Type =2 :Work Type =3 : Địa chỉ khác
7	Label	Text	Nhãn của dữ liệu.

Bảng 4.3: Cấu trúc bảng **Contact_Menthods** trong Cơ sở dữ liệu Contact.

Mặc định Android sử dụng bảng này để lưu trữ Email và Address tương ứng với Contact Item. Muốn lưu trữ các loại dữ liệu khác ta chỉ cần định nghĩa field **Kind**, miễn sao giá trị không trùng với các loại dữ liệu đã được định nghĩa sẵn trong Android.

Tóm lại, Android lưu trữ tất cả dữ liệu của Contact vào trong ba bảng như trên. Để truy xuất đến các dữ liệu này, ta phải sử dụng Content Provider.

4.1.2. Content Provider

Content Provider là một khái niệm mới trong Android, là một kỹ thuật cài đặt sẵn một tập các phương thức theo chuẩn nhằm giúp các ứng dụng khác nhau có thể chia sẽ dữ liệu với nhau.

Trong Android, dữ liệu lưu trữ dưới dạng cơ sỡ dữ liệu có thể được truy xuất thông qua các lớp đối tượng SQLite Database. Tuy nhiên, nếu truy cập vào cơ sỡ dữ liệu do ứng dụng khác tạo ra thì còn phục thuộc vào ứng dụng đó có cho phép hay không. Ví dụ, ứng dụng A muốn truy cập vào cơ sỡ dữ liệu do ứng dụng B tạo

ra, thì phụ thuộc vào ứng dụng B có cho phép các ứng dụng khác được truy cập vào cơ sỡ dữ liệu do mình tạo ra hay không. Mặc khác, nếu ứng dụng B cho phép thì các ứng dụng khác mặc nhiên được truy cập vào tất cả các thông tin lưu trong cơ sỡ dữ liệu của ứng dụng B.

Nhằm đáp ứng vấn đề cho các ứng dụng khác truy cập vào cơ sỡ dữ liệu nhưng hạn chế ở những dữ liệu nào cho phép, Android cung cấp khái niệm Content Provider đáp ứng vấn đề này. Lưu ý, Content Provider chỉ là phương pháp chuẩn để truy cập vào dữ liệu do các ứng khác tạo ra. Content Provider cài đặt một cú pháp cho phép yêu cầu truy xuất dữ liệu từ các ứng dụng khác.

Cũng giống như ngôn ngữ SQL,Content Provider cũng sử dụng chuỗi truy vấn để đọc ghi dữ liệu. Trong Content Provider, chuỗi truy vấn dữ liệu được gọi là chuỗi URI, và điểm khác biệt so với ngôn ngữ SQL là chuỗi URI diễn tả kiểu dữ liệu được trả về, cấu tạo gồm 3 thành phần sau:

- Bắt đầu : "**Content://**".
- Thành phần tiếp theo cho biết kiểu dữ liệu sẽ trả về.
- Thành phần cuối cùng là chỉ số ID , thành phần này có thể có hoặc không. Nếu có, thì trả về dòng đặt biệt tương ứng với ID của dữ liệu được trả về.

Ví dụ, chuỗi URI để lấy về dữ liệu People trong Cơ sở dữ liệu Contact trong Android:

content://contacts/people/.

Nếu thức hiện truy vấn với chuỗi URI trên thì kết quả trả về sẽ là :

_ID	_COUNT	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	4	(425)555 677	425 555 6677	California office	Bully Pulpit	Work
44	4	(212)555-1234	212 555 1234	NY apartment	Alan Vain	Home
45	4	(212) 555-6657	212 555 6657	Downtown office	Alan Vain	Work
53	4	201.555.4433	201 555 4433	Love Nest	Rex Cars	Home

Bảng 4.4: Kết quả trả về tương ứng với chuỗi truy vấn.

Còn với chuỗi truy vấn sau : **content://contacts/people/13** sẽ lấy về People có ID=13.

_ID	_COUNT	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	4	(425)555 677	425 555 6677	California office	Bully Pulpit	Work

Bảng 4.5: Kết quả trả về tương ứng với chuỗi truy vấn có id

Mội vài chuỗi URI được định nghĩa sẵn trong Android như sau :

Chuỗi URI	Ý nghĩa
content://contacts/people	Chuỗi URI tương ứng với bảng PEOPLE trong cơ sỡ dữ liệu Contact.
content://contacts/phone	Chuỗi URI tương ứng với bảng PHONES trong cơ sỡ dữ liệu Contact.
content://contacts/contact_menthods	Chuỗi URI tương ứng với bảng Contact_menthods trong cơ sỡ dữ liệu Contact.

Bảng 4.6: Các chuỗi URI định nghĩa sẵn trong Android.

4.1.3. Đọc dữ liệu Contact

Để đọc thông tin bảng PEOPLE , chúng ta sử dụng chuỗi URI như sau : **content://contacts/people/.** Với chuỗi URI này có thể đọc toàn bộ thông tin trong bảng PEOPLE. Còn để đọc thông tin chỉ của một Contact Item dựa vào ID thì có thể sử dụng chuỗi URI : **content://contacts/people/ID** (với ID tướng ứng với Content Item cần đọc dữ liệu)

Chú ý là các chuỗi URI dùng để truy vấn dữ liệu Contact đều được định nghĩa sẵn trong Android như sau :

Chuỗi URI	Dữ liệu trả về
Content://Contact/People	Lấy về thông tin của bảng PEOPLE
Content://Contact/Phone	Lấy về thông tin của bảng PHONE
Content://Contact/Contact_Methods	Lấy về thông tin của bảng Contact_Menthods

.Bảng 4.7: Các chuỗi URI truy vấn của Cơ sở dữ liệu Contact.

Để truy xuất dữ liệu tương ứng với các chuỗi URI như trên, ta sử dụng phương thức **query** của lớp **android.content.ContentResolver,** phương thức này trả về đối tượng **Android.Database.Cursol** chứa tập hợp dữ liệu đọc được tương ứng với chuỗi URI.

Ví dụ , để đọc thông tin PEOPLE trong Contact :

```
// Lấy về chuỗi URI đã được định nghĩa sẵn
Uri uriPeople = android.provider.Contacts.People.CONTENT_URI;
// truy vân dữ liệu.
Cursor cur = query(uriPeople, null, null, null, null);
```

Xử lý dữ liệu trả về : với đoạn lệnh như trên, dữ liệu trả về là tập hợp các record trong bảng PEOPLE của cơ sở dữ liệu Contact. Để đọc dữ liệu tương ứng với từng cột, ta làm theo các bước sau:

- Lấy về chỉ số cột cần đọc dựa vào tên cột.
- Đọc dữ liệu của cột thông quan chỉ số cột vừa tìm được.

Đoạn code sau sẽ lấy tất cả các tên (cột Name) trong bảng PEOPLE của cơ sỡ dữ liệu Contact.

```
Public ArrayList<String> getNames()
{
      ArrayList<String> arr = new ArrayList<Sting>();
      String NAME= android.provider.Contacts.People.Name; // "Name"
      Cursor rs = cr.query(uriPeople, null, null, null, null);
      if (rs.count()>0 && rs.first()){
            // Lấy về chỉ số của cột Name
            int nameIndex = rs.getColumnIndex(NAME);
            do{
                  String name = rs.getString(nameIndex);
                  arr.add(name);
            }while (rs.next());
      }
      rs.close();
      return arr;
}
```

Chú ý, tên của tất cả các cột tương ứng với các bảng trong cơ sở dữ liệu Contact đều đã được định nghĩa sẵn trong Android (trong gói **android.provider.Contacts**)

Bảng	Cột	Định nghĩa trong Android	Giá trị
People	_id	People._ID	"_ID"
	Name	People.NAME	"name"
	Notes	People.NOTES	"notes"
	Photo	People.PHOTO	"photo"
	Company	People.COMPANY	"company"
	Title	People.TITLE	"title"
	Times_contacted	People.TIMES_CONTACTED	"times_contacted"
	Last_time_contacted	People. LAST_TIME_CONTACTED	"last_time_contacted"
	Starred	People.STARRED	"starred"
	Preferred_phone	People.PREFERRED_PHONE	"preferred_phone"
	Preferred_email	People.PREFERRED_EMAIL	"preferred_email"
Phones	_id	Phones._ID	"_id"
	Person	Phones.PERSON_ID	"person"
	Type	Phones.TYPE	"type"
	Number	Phones.NUMBER	"number"
	Number_key	Phones.NUMBER_KEY	"number_key"
	Label	Phones.LABEL	"label"
Contact_menthods	_id	ContactMenthods._ID	"_id"
	Person	ContactMenthoids.PERSON_ID	"person"
	Kind	ContactMenthods.KIND	"kind"
	Data	ContactMenthods.DATA	"data"
	Aux_data	ContactMenthods.AUX_DATA	"aux_data"
	Type	ContactMenthods.TYPE	"type"
	Label	ContactMenthods.LABEL	"label"

Bảng 4.8: Định nghĩa các cột dữ liệu trong Cơ sở dữ liệu Contact.

4.1.4. Thêm mới Contact Item

Để thêm dữ liệu vào một Contact Item, chúng ta lần lượt thêm dữ liệu vào bảng PEOPLE, sau đó mới tiếp tục thêm dữ liệu vào bảng PHONES và CONTACT_MENTHODS (nếu có dữ liệu). Ví dụ để thêm tên và số điện thoại và trong CONTACT ta thực hiện tuần tự các bước như sau :

- Thêm dữ liệu **tên** vào trong bảng PEOPLE

```
ContentValues values = new ContentValues();
values.put(android.provider.Contacts.People.NAME, "Nguyen Duy Tan");
Uri uri = insert(android.provider.Contacts.People.CONTENT_URI, values);
// phương thức Insert của lớp ContentResolver
```

- Thêm dữ liệu **Số điện thoại** vào trong bảng PHONES vào Contact Item có ID tương ứng. Ví dụ thêm số điện thoại "0913141322" loại Mobile vào Contact Item có Id=5

```
Uri uriPhone =android.provider.Contacts.Phones.CONTENT_URI;
ContentValues values  =new ContentValues();
values.put("PERSON", 5);
values.put("NUMBER","0913141322");
values.put("TYPE",1);
// type =1 : số điện thoại Mobile
Uri uriRs = insert(uriPhone, values);
// phương thức Insert của lớp ContentResolver
```

- Để thêm các loại dữ liệu khác, ta có thể thêm vào trong bảng Contact_Menthods, ví dụ như Email hay Address, hoặc các loại dữ liệu khác do ta tự định nghĩa. Ví dụ, thêm Email : nguyendinhlehung@gmail.com vào trong Contact Item có ID=5

```
ContentValues values=  new ContentValues();
values.put("kind", 1); // kind =1 tương ứng với dữ liệu Email
values.put("data", "nguyendinhlehung@gmail.com");
values.put("person",5);
values.put("type", 1);
String uri = People.CONTENT_URI.toString()+"/"+ 5 + "/contact_methods/";
uriContactMenthods = Uri.parse(uri);
Uri rs = insert(uriContactMenthods, values);
// phương thức Insert của lớp ContentResolver
```

- Thêm dữ liệu Address vào trong bảng Contact_Menthods, ví dụ thêm dữ liệu "15 Bui Thi Xuan, Q1, TpHCM" tương ứng với Contact Item có ID=5

```
ContentValues values=  new ContentValues();
values.put("kind", 2); // kind =2 tương ứng với dữ liệu Address
values.put("data", " 15 Bui Thi Xuan, Q1, TpHCM");
values.put("person",5);
values.put("type", 1);
String uri = People.CONTENT_URI.toString()+"/" + 5 + "/contact_methods/";
uriContactMenthods = Uri.parse(uri);
Uri rs = insert(uriContactMenthods, values);
// phương thức Insert của lớp ContentResolver
```

- Để thêm các loại dữ liệu khác chưa được định nghĩa trong Contact, ta chỉ cần định nghĩa loại dữ liệu mới này tương ứng với cột **Kind** trong bảng Contact_Menthods **(**ví dụ : kind =3 tương ứng với dữ liệu địa chỉ Website). Và thêm dữ liệu này vào trong bảng Contact Menthods.

4.1.5. Cập nhập dữ liệu Contact

Để cập nhập dữ liệu Contact, ta sử dụng phương thức **update…** của lớp **Cursor** để cập nhập các giá trị tương ứng vào trong từng cột của dòng hiện hành. Sau đó, phải gọi phương thức **commitUpdates()** để hoàn tất cập nhập dữ liệu.

Ví dụ, thay đổi tên của Contact Item có ID=5 thành tên "Duy Tan"

```
// lấy về chuỗi Uri tương ứng với dữ liệu cần cập nhập
String strUri = People.CONTENT_URI.toString() + "/" + 5;
Uri uri = Uri.parse(strUri);
//truy vấn dữ liệu
Cursor rs = cr.query(uri, null, null, null, null);
rs.first();
//lấ về chỉ số cột NAME
int nameColIndex = rs.getColumnIndex("name")
//cập nhập dữ liệu
rs.updateString(nameColIndex, "Duy Tan");
rs.commitUpdates();
```

4.1.6. Xóa Contact

Để xóa dữ liệu, ta gọi phương thức **deleteRow** của lớp **Cursor** để xóa dòng hiện hành. Sau đó, gọi phương thức **commitUpdates()** để xác nhận thao tác xóa

Ví dụ , đoạn code sau sẽ xóa Contact Item có ID=5

```
// lấy về chuỗi Uri tương ứng với dữ liệu cần cập nhập
String strUri = People.CONTENT_URI.toString() + "/" + 5;
Uri uri = Uri.parse(strUri);
//truy vấn dữ liệu
Cursor rs = cr.query(uri, null, null, null, null);
rs.first();
// xóa dòng dữ liệu
rs.deleteRow();
rs.commitUpdates();
```

4.1.7. Lưu trữ các thông tin khác trong cơ sờ dữ liệu Contact.

Android đã qui định sẵn cơ sỡ dữ liệu Contact để lưu trữ thông tin của người dùng. Do đó, không thẻ thay đổi được cấu trúc cơ sỡ dữ liệu Contact. Vấn đền đặt ra liệu có giải pháp nào để lưu trữ thêm các thông tin khác (ngoài những thông tin mà Android hỗ trợ) vào trong cơ sỡ dữ liệu Contact hay không.

Như đã nói trong phần 4.1.1 (mô hình lưu trữ Contact trong Android), Android cho phép người phát triển lưu trữ các thông khác vào trong bảng Contact_Menthods mà vẫn đảm bảo tương thích với các ứng dụng khác.

Ví dụ, để lưu trữ thêm khóa công khai (public key) vào trong một Contact Item được thực hiện như sau:

- Bước 1: định nghĩa loại dữ liệu cần lưu tương ứng với một số nguyên để lưu trữ trong vào bảng Contact_Menthods ở field **kind**. Giá trị này khác với các giá trị đã được định nghĩ sẵn trong Android (Giá trị định nghĩa sẵn trong Android : 1 – tương ứng với dữ liệu Email, 2 – tương ứng với dữ liệu Address).
- Bước 2: dữ liệu cần lưu sẽ được lưu trữ tương ứng trong field **data** của bảng Contact_Menthods

Tương ứng với ví dụ trên, thông tin khóa công cộng (public key) sẽ được lưu trữ trong field **data** tương ứng field **kind** có giá trị bằng **4**.

_id	Person	Kind	data	Aux_data	Type	Label
7	1	**4**	**abcdmisfhfiaHfuu.**		0	**Public Key**

Bảng 4.9: Ví dụ lưu trữ thông tin khóa công khai trong bảng Contact_Menthods

Ngoài ra, trong phạm vi ứng dụng sẽ xây dựng sẽ sử dụng bảng Contact_Menthods để lưu trữ những thông tin sau:

Loại dữ liệu	Giá trị định nghĩa cho field **kind**	Ý nghĩa
IsEncrypt	3	Đánh dấu Contact Item đã mã hóa hay chưa
Public Key	4	Lưu trữ thông tin khóa công khai của một Contact Item
Secret Key	5	Lưu trữ thông tin khóa bí mật được dùng trong quá trình mã hóa tin nhắn (phần này trình bày trong chương 5)

Bảng 4.10: Các thông tin khác được lưu trữ trong Contact

4.1.8. Mã hóa và lưu trữ thông tin Contact

Mã hóa và lưu trữ thông tin Contact là việc thay đổi các thông tin của Contact Item với các giá trị đã được mã hóa. Vấn đề làm sao lưu trữ các giá trị đã mã hóa vào trong cơ sỡ dữ liệu Contact. Các giá trị sau khi mã hóa có trị bất kỳ trong khoảng từ 0 đến 255. Mặt khác, cơ sở dữ liệu trong Android không thể lưu được byte có giá trị bằng 0. Để giải quyết vấn đề này, giá trị sau khi mã hóa sẽ được chuyển qua kiểu **Base64** trước khi lưu trữ vào trong cơ sở dữ liệu Contact.

4.2. Gọi Activity khác trong cùng một ứng dụng

4.2.1. Vấn đề

Trong cùng một ứng dụng, có hai Activity A và B. Mỗi activity thực hiện một công việc khác nhau của ứng dụng. Vấn đề, từ Activity A truyền dữ liệu và gọi thực hiện Activity B. Sau đó nhận kết quả trả về sau khi Activity B xử lý xong.

Ví dụ: từ Activity **ViewDetailContact** ta gọi Activity **EditContact** để cập nhập dữ liệu. Sau khi activity **EditContact** thực hiện xong, ta phải cập nhập lại dữ liệu trong activity **ViewContact**

4.2.2. Giải pháp

- Trong ứng dụng có bao nhiêu activity thì phải khai báo tất cả trong tập tin cấu hình AndroidManifesh.XML.
- Khởi tao đối tượng Intent với tham số tên lớp activity cần gọi thực hiện
- Khởi tao đối tượng Bundle để truyền đối số thích hợp để Activity cần gọi thực hiện xử lý.
- Truyền các đối số thích hợp vào đối tượng Bundle.
- Đặt đối tượng Bundle vào đối tượng Intent thông qua phương thức putExtras.
- Gọi thức hiện Activity thông qua phương thức startSubActivity của lớp activity.
- Xử lý sự kiện **onActivityResult()** khi activity cần gọi đã thực hiện xong.

4.2.3. Chi tiết giải pháp

- Khai báo các Activity trong file cấu hình ứng dụng AndroidManifesh.XML.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="gpcontact.ui">
    <application android:icon="@drawable/icon">
       <activity android:name=".ViewContat" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".EditContact" android:label="@string/app_name/">
</application>
</manifest>
```

- Gọi thức hiện Activity EditContact, truyền tham số id của Contact Item cần cập nhập cho activity EditContact cần thực hiện.

```
// khởi tạo đối tượng intent để di chuyển qua lai các màn hình
Intent i = new Intent(this,EditContact.class);
//khởi tạo đối tượng Bundle để truyền các đối số cho activity EditContact
Bundle wr = new Bundle();
//truyên đối số isEdit và idContacts để EditContac thực hiện cập nhập dữ
liệu với Contact Item có id=3
wr.putBoolean("isEdit",true");
wr.putInt("idContacts",3);
// Đặt đối tượng Bundle này vào đối tượng Intent
i.putExtras(wr);
// Gọi thực hiện activity với tham số request code
startSubActivity(i, requestcode);  // requestcode=1
//nếu tham số requestcode >0 thì sự kiện OnActivityresult sẽ dược gọi
ngay sau khi activity thực hiện xong
```

- Xử lý sự kiện onActivityResult() trong activity ViewContac sau khi activity EditContact đã thực hiện xong

```
@Override
protected void onActivityResult(int requestCode, int resultCode, String
data, Bundle extras)
{
      super.onActivityResult(requestCode, resultCode, data, extras);
      if (requestCode == 1)
      {

      }
 }
```

4.3. Gọi Activity giữa hai ứng dụng khác nhau

4.3.1. Vấn đề

Ứng dụng A muốn gọi thực hiện một thao tác nào đó của ứng dụng B, nghĩa là muốn gọi một Activity của ứng dụng B vào thời điểm Runtime. Ví dụ như từ ứng dụng Contact gọi chức năng gởi tin nhắn của ứng dụng SMS.

4.3.2. Giải pháp

Trong tập tin cấu hình AndroidManifesh.XML của ứng dụng SMS, khai báo activity Create_Message để các ứng dụng khác có thể gọi thực hiện activity này.

```
<activity android:name=".Create_message">
   <intent-filter>
      <action   android:name="android.intent.action.Create_Message"/>
      <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
 </activity>
```

Dĩ nhiên Activity **Create_message** phải được khái báo trong ứng dụng để các Activity trong cùng ứng dụng có thể gọi nó. Ngoài ra, để các ứng dụng khác có thể truy cập vào nó, ta phải khai báo thêm tag **<intent-filter>** trong tag <activity>, tag này gồm hai tag con **<action> và <category>**.

Trong cấu hình trên, tag <action> là quan trọng nhất, phải khái báo thuộc tính **android:name** của tag này có giá trị duy nhất trong hệ thống Android.. Với khai báo như trên, thì ứng dụng sẽ đăng ký với hệ thống Android một Action có tên là : **android.intent.action.Create_Message**, để bất cứ ứng dụng nào cũng có thể gọi thực hiện Activity Create_Message.

Trong các ứng dụng khác, để thực hiện activity Create_Message, ta thực hiện như sau :

```
// khởi tạo một Intent với tên Activity đã đăng ký trong hệ thống
Intent i = new Intent("android.intent.action.Create_Message");
// gọi thực hiện activity này với phương thức startSubAntivity()
startSubActivity(I, 0);
```

4.3.3. Chi tiết giải pháp

- Cấu hình trong tập tin AndroidManifesh.XML của ứng dụng SMS

```
<activity android:name=".Create_message">
   <intent-filter>
      <action   android:name="android.intent.action.Create_Message"/>
      <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
 </activity>
```

- Khởi tao Intent và gọi thực hiện Activity Create_Message

```
// khởi tao intent để gọi thực hiện activity
Intent i = new Intent("android.intent.action.Create_Message");
//khởi tạo đối tượng Bundle để truyền đối số cho activity Create_Message
Bundle wr = new Bundle();
wr.putString("phoneNumber","0913141322"); // số điện thoại cần gởi tin
nhắn
i.putExtras(wr);
//gọi thực hiện Activity
this.startSubActivity(i, 0);
```

4.4. Kỹ thuật xử lý cuộc gọi

4.4.1. Vấn đề

Trong ứng dụng Contact, cho phép gọi nhanh một số điện thoại có trong Contact.

4.4.2. Giải pháp

Thao tác thực hiện một cuộc gọi được xây dựng sẵn trong hệ thống Android.

Hình 4.2: Màn hình gọi điện thoại

Chúng ta chỉ cần gọi thực hiện Activity này từ trong ứng dụng Contact. Như vậy, quay lại vấn đế gọi thực hiện activity giữa hai ứng dụng khác nhau. Ở đấy, đặt biệt, activity dùng để gọi điện thoại được đăng ký sẵn trong Android với ACTION là : **android.intent.action.CALL** và khi gọi thực hiện activity này thì phải cho biết nó thực hiện với Contact Item nào trong Contact. Ngoài ra, cần phải đăng ký ứng dụng được phép gọi điện thoại với hệ thống trong file cấu hình AndroidManifesh.XML

4.4.3. Chi tiết giải pháp

Để thực hiện một cuộc gọi, lần lược thức hiện các bước sau :

- Đăng ký ứng dụng được gọi điện thoại với hệ thống Android trong file **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="khtn.gpcontact">
   <uses-permission android:name="android.permission.CALL_PHONE" />
   <application android:icon="@drawable/icon">
    ...
   </application>
</manifest>
```

- Xác định ID của số điện thoại cần gọi , ví dụ ID=5
- Khởi tạo Intent với Action là Intent.CALL_ACTION và dữ liệu là chuỗi URI: **content://contacts/phones/5**
- Gọi thực hiện Intent với phương thức startSubActivity của lớp Activity.

```
Uri data = Uri.parse(Phones.CONTENT_URI.toString() + "/" +5)
Intent is = new Intent(Intent.CALL_ACTION,data);
startSubActivity(is, 0);
```

- Ngoài ra còn có thể thực hiện cuộc gọi dựa vào ID của Contact Item, khi đó, mặc nhiên sẽ gọi thực hiện cuộc gọi tương ứng với số điện thoại chính của Contact Item đó. Ví dụ, gọi điện thoại với Contact Item có ID =5

```
Uri data = Uri.parse(People.CONTENT_URI.toString() + "/" +5)
Intent is = new Intent(Intent.CALL_ACTION,data);
startSubActivity(is, 0);
```

4.5. Kỹ thuật xử lý tin nhắn

4.5.1. Giới thiệu

Trong Android chưa có hệ thống xử lý và lưu trữ tin nhắn, mà chỉ mới cung cấp các hàm API để gởi và nhận tin nhắn.

Tin nhắn gởi đến được thực hiện thông qua việc giả lập từ công cụ được cung cấp kèm theo SDK, còn việc gởi tin nhắn cung cấp qua hàm API.

4.5.2. Đặc điểm tin nhắn trong Android

Tin nhắn trong Android có những đặc điểm sau:

- Hỗ trợ các dạng encoding sau : 7 bit, 8 bit, 16 bit
- Mặc định việc nhận tin nhắn từ công cụ giả lập chỉ thực hiện được với tin nhắn 7bit.
- Theo như tài liệu của Google, thì tương ứng với dạng SMS 7bit thì số ký tự tối đa cho mỗi tin nhắn là 160 ký tự. Nhưng khi khảo sát thì việc gởi tin nhắn từ công cụ đến máy ảo thì có kết quả như sau :

STT	Nội dung	Chiều dài	Kết quả
1	Chuỗi rỗng	0	- không nhận được. Chưa biết nguyên nhân công cụ giả lập hay do máy ảo không nhận tin nhắn không có nội dung
2	Chuỗi ký tự	<=154	Nhận được một tin nhắn với đầy đủ nội dung Không có header
3	Chuỗi ký tự	>154	Nhận được số tin nhắn như sau : (chiều dài / 155) + 1 Với mỗi tin nhắn có kèm thêm Header.

Bảng 4.11: Khảo sát cấu trúc của tin nhắn.

- Như vậy hiện tại chỉ giả lập được một tin nhắn có chiều dài không quá 154..
- Với tin nhắn nhỏ hơn 154 ký tự thì không nhận được header. Ngược lại, nhận về header có kèm các thông tin sau :

Byte	Tên	Ý nghĩa
1	SM Reference Number	Các tin nhắn con được gởi từ cùng một tin nhắn có độ dài lớn hơn 160 ký tự sẽ có cùng giá trị này.
2	Numer of Mersseners	Số lượng tin nhắn
3	This SM Sequence Numbers	Số thứ tự của tin nhắn con này trong tin nhắn lớn.

Bảng 4.12 Thông tin header.

Tin nhắn SMS được Android bao bọc trong lớp **SmsMessage (trong gói android.Telephony.gsm)**, với một vài hàm thông dụng để lấy về nội dung tin nhắn và số điện thoại gởi đến.

Ngoài ra, nếu có nhiều tin nhắn con được gởi đến từ một tin nhắn lớn (có chiều dài lớn hơn 160 ký tự) thì mỗi tin nhắn con sẽ có thêm header gởi kèm theo mỗi tin nhắn. header có thể được sử dụng cho nhiều mục đích khác nhau. Và tùy từng vào mục đích sử dụng. header sẽ có nội dung và cấu trúc khác nhau. Trong Android, header được bao bọc trong lớp **SmsHeader** (**android.Telephony.gsm).**

4.5.3. Nhận tin nhắn gởi đến

Android coi tin nhắn gởi đến điện thoại là một tác động bên ngoài đến điện thoại (giống như các tác động khác như cuộc gọi đến…). Android cung cấp lớp IntenReceiver để bắt các sự kiện bên ngoài tác động đến điện thoại. Bản thân IntentReceiver không thể hiện một giao diện cụ thể nào cho người dùng, và các ứng dụng có chứa IntentReceiver không cần phải chạy. Ứng dụng chỉ cần đăng ký IntentReceiver mà mình muốn xử lý, thì khi có IntentRecever được gởi hệ thống thì hệ thống sẽ gọi thực thi ứng dụng.

Để thực hiện nhận tin nhắn, trước hết phải đăng ký nhận tin nhắn với hệ thống. để thực hiện điều này, phải cấu hình trong file AndroidManifest.XML như sau :

```
<receiver android:name=".SMSReceiver">
    <intent-filter>
       <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

Sau đó, phải tạo một lớp kế thừa từ lớp IntentReceiver. Lớp này được dùng để đón bắt các sự kiện khi có tác động từ bên ngoài gởi đến. Override sự kiện OnReceiveIntent (sự kiện này xảy ra khi có một tác động từ bên ngoài đến điện thoại)

Đoạn code sau sẽ cho biết tác động từ bên ngoài có phải là tin nhắn đến hay không, sau đó lấy về tất cả tin nhắn gởi đến.

```
String ACTION = "android.provider.Telephony.SMS_RECEIVED";
Class SMSReceiver extends IntentReceiver {
 public void onReceiveIntent(Context context, Intent intent)
 {
  // Nếu Intent nhận được là nhận được tin nhắn
  if (intent.getAction().equals(ACTION))
  {
    if (bundle != null) {
    // Lấy về tất cả các tin nhắn có trong Intent này
    SmsMessage[] mess=Telephony.Sms.Intents.getMessagesFromIntent(intent)
    }
  }
}
```

4.5.4. Gởi tin nhắn

Thực hiện gởi tin nhắn trong Andoid được thực hiện thông qua lớp SMSManager (android.telephony.gsm). Lớp này bào gồm các phương thức dùng cho việc gởi dữ liệu, tin nhắn.Bao gồm các phương thức thông dụng sau :

Tên phương thức	Ý nghĩa
divideMessage(String text)	Phân tích một tin nhắn thành một dãy các tin nhắn con có chiều dài không vượt quá chiều dài lớn nhất của tin nhắn.
sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, Intent sentIntent, Intent deliveryIntent, Intent failedIntent)	Hàm thực hiện gởi dữ liệu dựa trên tin nhắn SMS.
sendTextMessage(String destinationAddress, String scAddress, String text, Intent sentIntent, Intent deliveryIntent, Intent failedIntent)	Hàm thực hiện gởi văn bản dựa trên tin nhắn SMS.

Bảng 4.13 Các phương thức lớp SmsManager

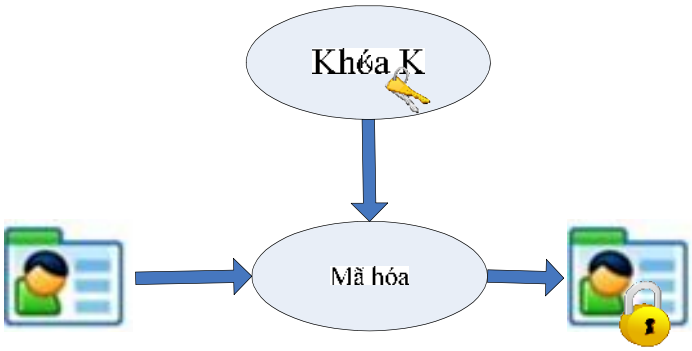
4.5.5. Mã hóa và giải mã tin nhắn

Mã hóa tin nhắn SMS: mục đích của việc mã hóa tin nhắn là nhằm bảo vệ nội dung của tin nhắn do đó chỉ quan tâm đến phần nội dung của tin nhắn mà không phải thay đổi các thông số đã có trên tin nhắn

Việc mã hóa tin nhắn SMS gặp một vấn đề như sau: mạng dịch vụ gởi tin nhắn GSM chỉ hỗ trợ các ký tự 7 bit vì thế bộ mã ASCII sau khi mã hóa và gởi đi sẽ có một số ký tự bị mất. Để giải quyết vấn đề này ta phải dùng hệ chuyển đổi ký tự 8 bit thành ký tự 6 bit. Các ký tự sau khi chuyển đổi có thể gởi qua mạng GSM.

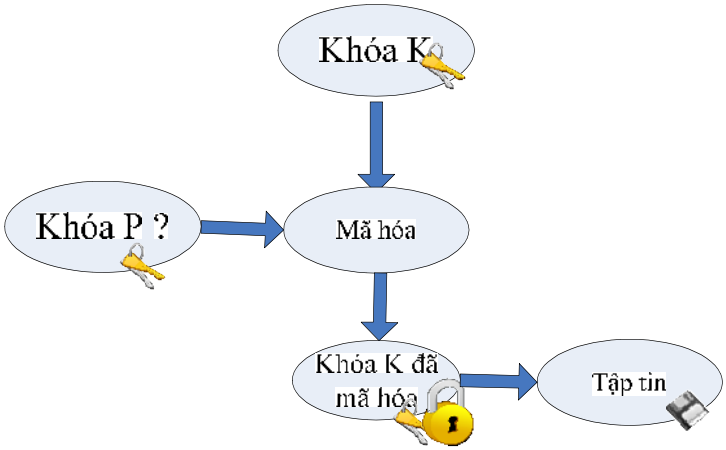
4.6. Kỹ thuật lưu trữ khóa

Khóa được sử dụng cho mã hóa và giải mã các thông tin (thông tin liên lạc trong Contact, hay tin nhắn SMS) cần được bảo mật. Do đó, thông tin khóa cũng cần được bảo mật. Khóa không thể lưu trữ xuống tập tin theo cách thông thường, mà phải được mã hóa trước khi lưu trữ xuống tập tin.. Bởi vì, các thông tin đã mã hóa hoàn toàn có thể giải mã được nếu để lộ khóa.



Hình 4.3: Sử dụng khóa mã hóa thông tin

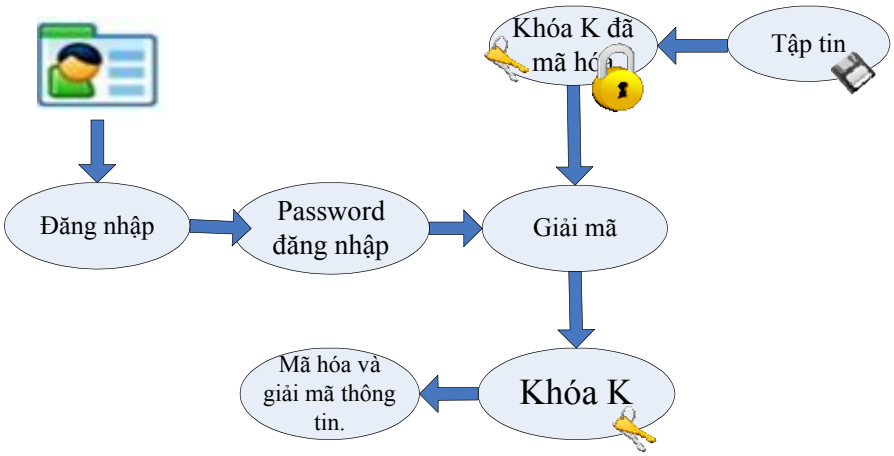
Tuy nhiên, vấn đề đặt ra, khi mã hóa khóa K (khóa dùng để mã hóa thông tin Contact hay SMS), chúng ta lại cần một khóa P khác. Vậy khóa P lấy ở đâu ?.



Hình 4.4: Mã hóa khóa K.

Vấn đề này có thể giải quyết bằng cách để người dùng tự quyết định khóa **P,** nghĩa là khi đang nhập vào ứng dụng, người dùng phải nhập vào khóa **P (**dưới dạng mật khẩu đang nhập ứng dụng) để mã hóa và giải mã khóa **K** (khóa dùng để mã

hóa và giải mã thông tin). Như vậy, các bước lấy khóa đã mã hóa dưới tập tin dùng để mã hóa và giải mã thông tin được tóm tắt lại như sau:



Hình 4.5: Giãi mã khóa từ tập tin.

- Bước 1: người dùng đăng nhập, kiểm tra đăng nhập. Nếu đăng nhập thành công chuyển sang bước 2.
- Bước 2: lấy password đã đăng nhập của người dùng làm khóa **P**.
- Bước 3: lấy khóa **P** giải mã tập tin chứa khóa **K.** Khóa K này sẽ được sử dụng để mã hóa vá giải mã thông tin trên điện thoại.

4.7. Kết luận

Trong chương này chúng em trình bày về một số tìm hiểu về các kỹ thuật cần thiết về Contact, SMS và các kỹ thuật khác để xây dựng các ứng dụng như Contact hay SMS trên nền tảng Android. Trong chương 5 sẽ trình bày các quy trình bảo mật thông tin Contact và SMS trên môi trường Android.

Chương 5

Một số quy trình đề nghị để bảo mật thông tin trên điện thoại di động

✍ *Nội dung của chương 5 trình bày một số quy trình mà nhóm đã xây dựng nhằm thiết lập chế độ bảo mật dữ liệu trong Android. Các quy trình chính mà nhóm đã xây dựng bao gồm:*

- *Một số quy trình đề nghị để bảo mật thông tin liên lạc (contact) trong Android.*
- *Quy trình bảo mật tin nhắn SMS trong Android.*
- *Quy trình trao đổi khóa.*

5.1. Quy trình mã hóa trong ứng dụng Contact

5.1.1. Giới thiệu quy trình mã hóa

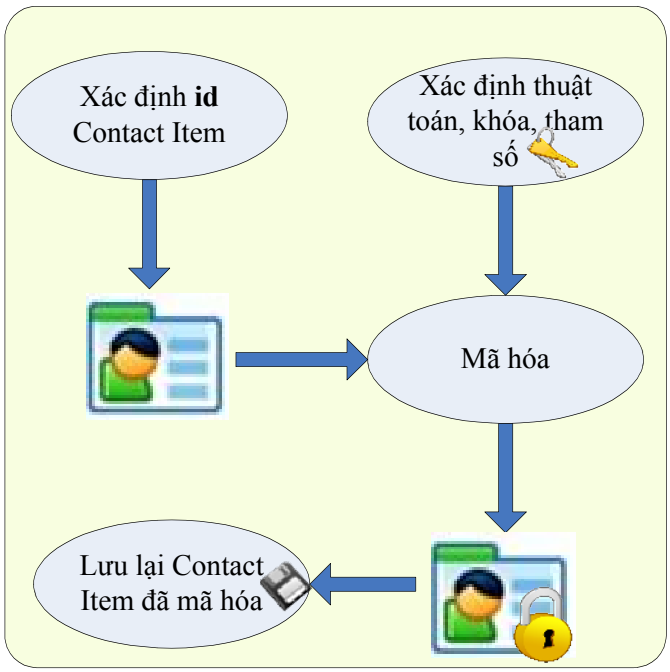
Quy trình mã hóa và giải mã thông tin Contact trong Android bao gồm các phần chính sau :

- Mã hóa thông tin Contact đã có sẵn
- Mã hóa thông tin Contact trong quá trình thêm mới
- Giải mã thông tin Contact
- Giải mã thông tin Contact tương ứng khi có cuộc gọi đến
- Gọi điện thoại với thông tin Contact tương ứng đã bị mã hóa

5.1.2. Mã hóa Contact Item đã có sẵn

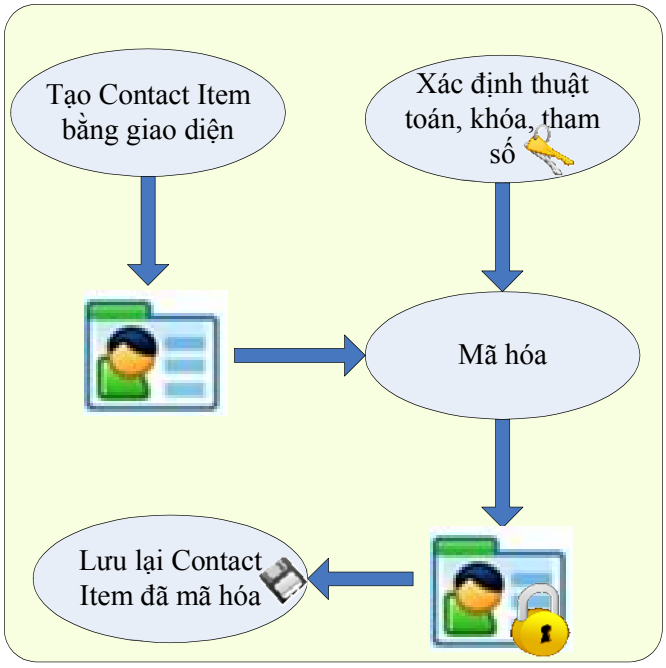
Các bước để mã hóa một Contact Item đã có sẵn trong cơ sở dữ liệu Contact của Android:

- Bước 1: Xác định Id của Contact Item cần mã hóa
- Bước 2: Xác định thuật toán, tham số và khóa liên quan đến thuật toán đã chọn để mã hóa
- Bước 3: Đọc nội dung Contact dựa vào Id và tiến hành mã hóa từng thông tin có trong Contact Item đó.
- Bước 4: Lưu trữ thông tin Contact Item này xuống cơ sở dữ liệu Contact.



Hình 5.1 : Quy trình mã hóa Contact Item đã có sẵn.

5.1.3. Mã hóa Contact Item mới được tạo ra

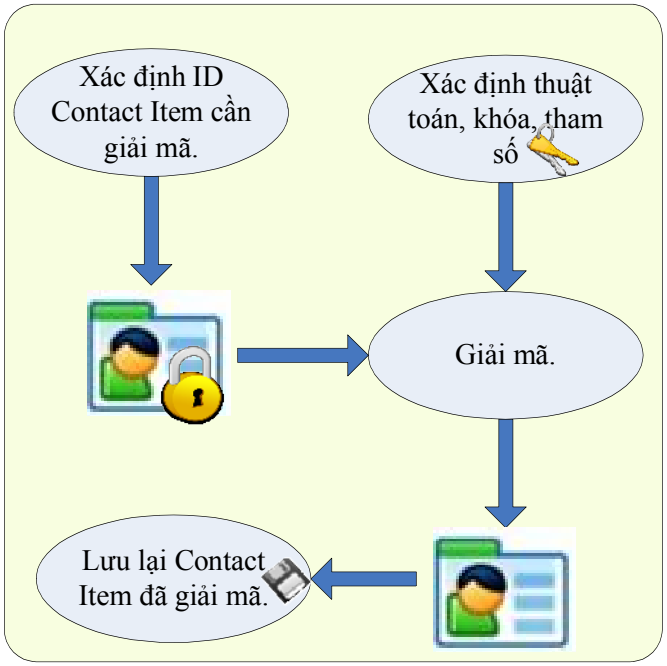


Hình 5.2 : Quy trình mã hóa Contact Item mới được tạo ra.

Các bước mã hóa trong quá trình thêm mới một Contact:

- Bước 1: Tạo một Contact Item từ giao diện của ứng dụng Contact.
- Bước 2: Xác định thuật toán, tham số và khóa liên quan đến thuật toán đã chọn để mã hóa
- Bước 3: Mã hóa từng thông tin có trong Contact Item
- Bước 4: Lưu trữ thông tin Conact vừa mới được tạo, đã mã hóa xuống cơ sở dữ liệu Contact.

5.1.4. Giải mã Contact Item đã mã hóa



Hình 5.3 : Quy trình giải mã thông tin Contact Item.

Các bước giãi mã một Contact Item đã bị mã hóa trong cơ sở dữ liệu Contact

- Bước 1: Xác nhận ID của Contact Item cần giải mã.
- Bước 2: Xác nhận Contact Item này đã mã hóa hay chưa, nếu chưa mã hóa thông báo và không làm gì thêm.
- Bước 3: Xác định các thuật toán và các tham số liên quan đến các thuật toán đã mã hóa Contact Item này.
- Bước 4: Đọc nội dung Contact Item tương ứng với Id và giải mã từng thông tin có trong Contact Item này.
- Bước 5: Lưu trữ Contact Item vừa được giải mã xuống cơ sở dữ liệu Contact.

5.1.5. Gọi điện thoại tương ứng với Contact Item đã mã hóa

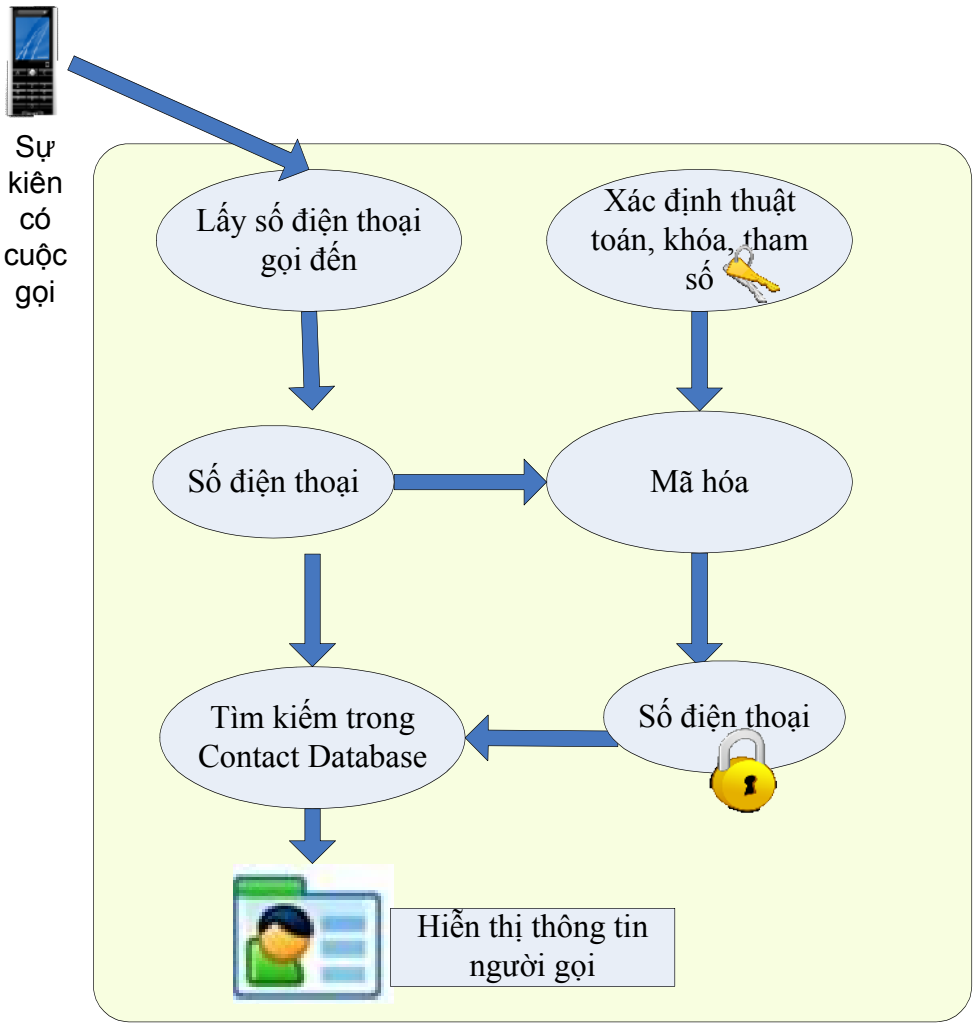
Đối với Contact Item đã bị mã hóa, trước khi thực hiện cuộc gọi từ giao diện ứng dụng, phải giải mã thông tin Contact Item để lấy về thông tin số điện thoại nhằm thực hiện cuộc gọi. Các bước thực hiện cuộc gọi tương ứng với Contact Item đã mã hóa :

- Bước 1: Xác định ID của Contact Item cần gọi điện thoại.
- Bước 2: Xác định khóa vá các tham số liên quan đến thuật toán đã mã hóa Contact Item.
- Bước 3: Giải mã Contact Item dựa vào các tham số đã xác định.
- Bước 4: Thực hiện gọi điện thoại tương ứng số điện thoại vừa giải mã được.

5.1.6. Giải mã Contact Item tương ứng với cuộc gọi đến

Khi có cuộc gọi đến, nếu số điện thoại đến tương ứng với Contact Item đã bị mã hóa trong cơ sở dữ liệu Contact, thì phải giải mã thông tin Contact Item này để hiển thị thông tin người gọi đến. Các bước giãi mã Contact Item tương ứng với cuộc gọi đến:

- Bước 1: Bắt sự kiện có cuộc gọi đến và lấy về thông tin số điện thoại.
- Bước 2: Xác định khóa và tham số để mã hóa số điện thoại gọi đến, dùng để tìm kiếm cho bước 3.
- Bước 3: Tìm kiếm số điện thoại (đầu vào của bước này bao gồm số điện thoại chưa mã hóa và đã mã hóa) gọi đến có trong cơ sở dữ liệu Contact .
- Bước 4: Nếu tìm thấy xác định Contact Item chứa số điện thoại gọi đến thì hiển thị thông tin người gọi.



Hình 5.4 : Quy trình giải mã Contact Item tương ứng với cuộc gọi đến

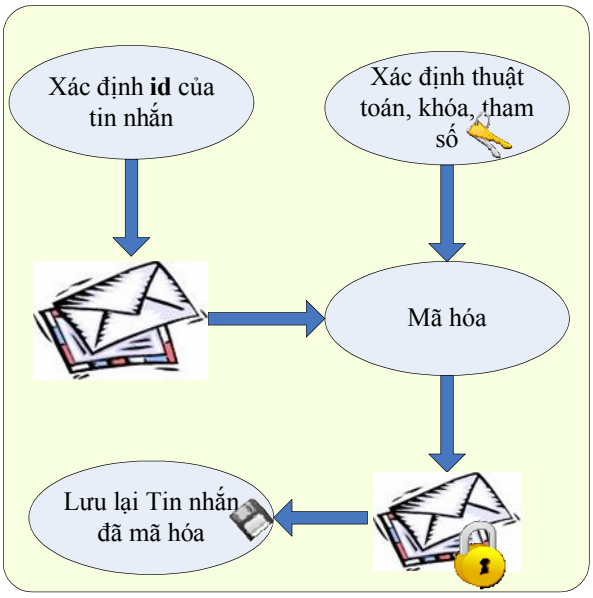
5.2. Quy trình mã hóa tin nhắn Sms

5.2.1. Giới thiệu quy trình mã hóa.

Trên các dòng điện thoại di động nói chung, và trong hệ điều hành Android nói riêng, quy trình bảo mật tin nhắn SMS bao gồm những phần như sau :

- Mã hóa tin nhắn đã có sẵn trong điện thoại.
- Mã hóa tin nhắn vừa được gởi đến.
- Giải mã tin nhắn đã được mã hóa.
- Gởi và nhận tin nhắn được mã hóa theo thuật toán đối xứng.
- Quy trình trao đổi khóa bằng tin nhắn SMS.

5.2.2. Mã hóa tin nhắn đã có sẵn trong điện thoại di động



Hình 5.5: Quy trình mã hóa tin nhắn có sẵn trong điện thoại

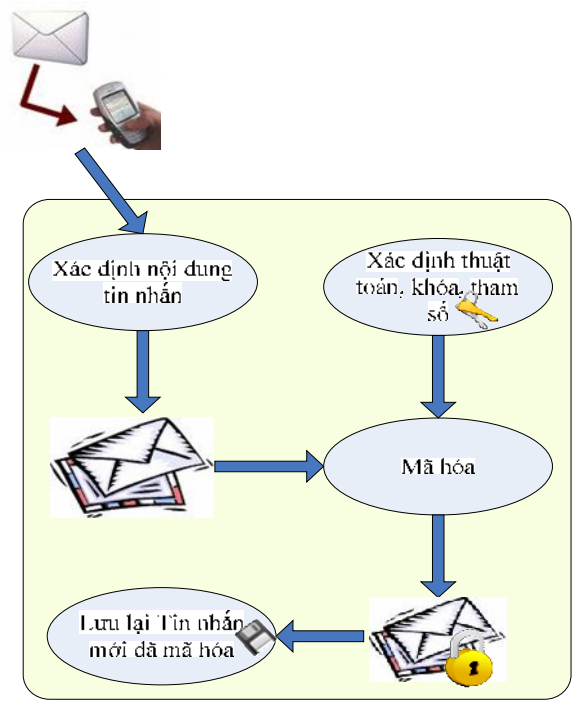
Các bước thực hiện mã hóa tin nhắn đã có sẵn trong điện thoại di động:

- Bước 1: Xác nhận Id của tin nhắn cần mã hóa.
- Bước 2: Xác nhận tin nhắn cần mã hóa là tin nhắn gốc hay tin nhắn đã được mã hóa.
- Bước 3: Xác nhận thuật toán và các tham số liên quan đến thuật toán cần mã hóa.
- Bước 4: Mã hóa nội dung tin nhắn dựa trên thuật toán và tham số đã chọn.
- Bước 5: Lưu lại tin nhắn với nội dung đã được mã hóa.

5.2.3. Mã hóa tin nhắn vừa nhận được.

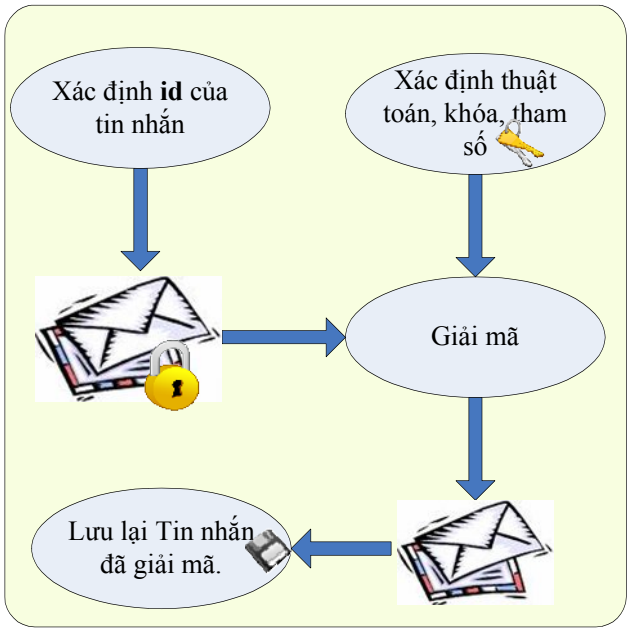
Mã hóa tin nhắn vừa nhận được trong Android, bao gồm các bước như sau:

- Bước 1: Xác định nội dung tin nhắn được gởi đến (phần này được trình bay chi tiết trong chương 4).
- Bước 2: Xác nhận thuật toán và các tham số liên quan đến thuật toán cần mã hóa.
- Bước 3: Mã hóa nội dung tin nhắn dựa trên thuật toán và tham số đã chọn.
- Bước 4: Tạo một tin nhắn mới với nội dung đã được mã hóa và lưu trữ xuống hệ thống lưu trữ tin nhắn.



Hình 5.6: Quy trình giải mã hóa tin nhắn mới đến.

5.2.4. Giải mã tin nhắn



Hình 5.7: Quy trình giải mã tin nhắn

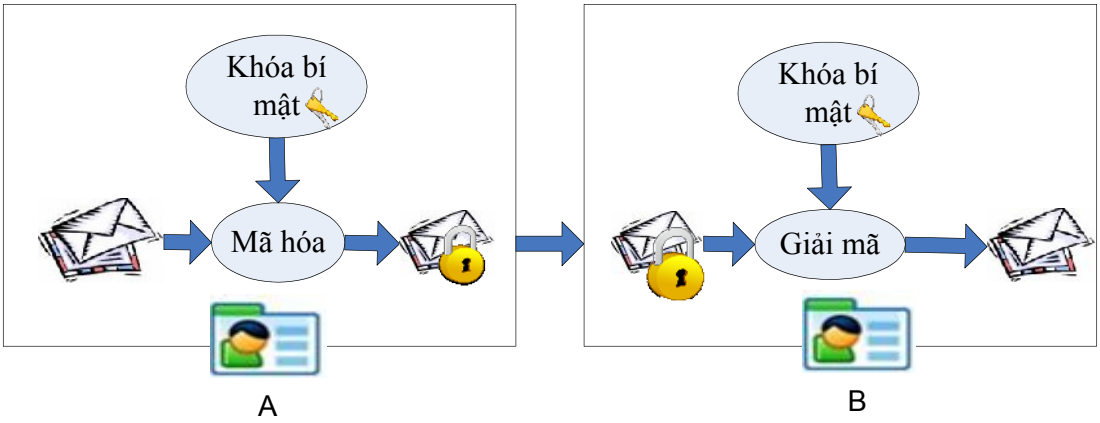
Các bước giải mã tin nhắn bao gồm các bước theo thứ tự thực hiện như sau:

- Bước 1: xác nhận id của tin nhắn cần giải mã.
- Bước 2: xác nhận nội dung của tin nhắn đã được mã hóa hay chưa.
- Bước 3: xác nhận thuật toán đã dùng để mã hóa tin nhắn, xác định khóa và các tham số liên quan đến thuật toán để giải mã tin nhắn.
- Bước 4: giải mã nội dung của tin nhắn với các thông số được xác định ở bước 3.
- Bước 5: lưu lại tin nhắn với nội dung đã được giải mã.

5.2.5. Quy trình trao đổi khóa bí mật bằng tin nhắn SMS

Nhằm đảm bảo nội dung tin nhắn được bảo mật trong quá trình trao đổi của người dùng, tin nhắn phải được mã hóa trước khi gởi và chỉ được giải mã sau khi nhận. Do đó, trong quá trình gởi nội tin nhắn đảm bảo bí mật và an toàn. Quy trình này gồm các bước như sau:

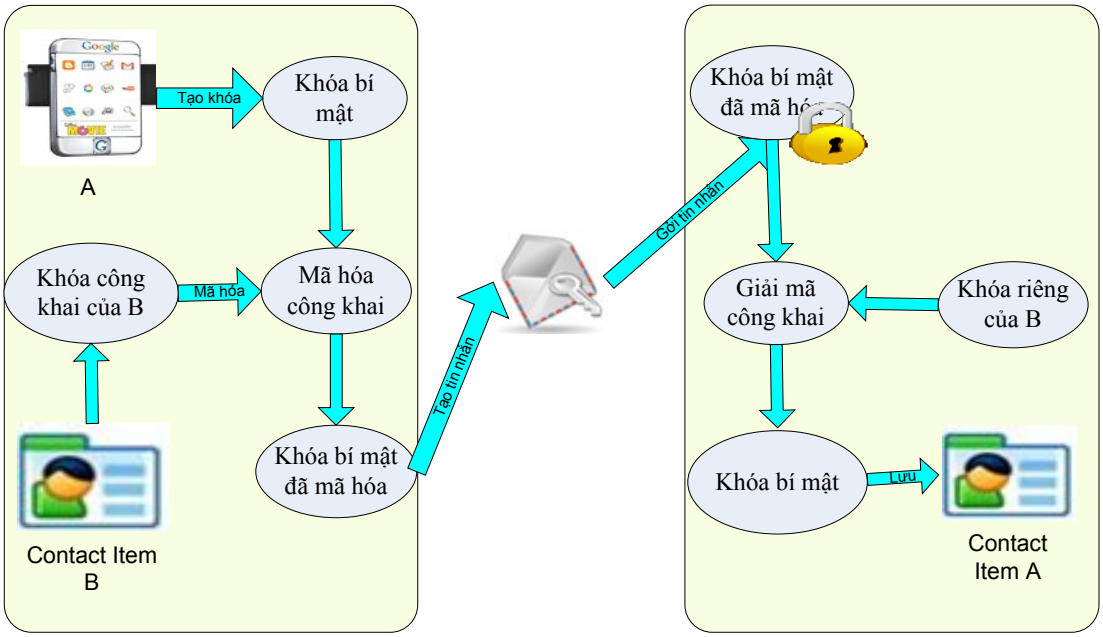
- Bước 1: A và B thỏa thuận khóa bí mật và các tham số dùng để mã hóa và giải mã tin nhắn trước và sau khi nhận.
- Bước 2: A tạo nội dung tin nhắn cần gởi và mã hóa bằng khóa bí mật và các tham số đã thoả thuận trước đó.
- Bước 3: B nhận được tin nhắn, giải mã nội dung tin nhắn bằng khóa bí mật và các tham số đã thỏa thuận trước đó.



Hình 5.8: Mô hình gởi và nhận tin nhắn SMS được mã hóa.

Trước khi gởi tin nhắn, hai bên gởi và nhận phải thống nhất trước khóa bí mật để mã hóa và giải mã tin nhắn. Quá trình thỏa thuận hay trao đổi khóa có thể diễn ra trên nhiều kênh thông tin khác nhau như điện thoại, email… hoặc trao đổi khóa thông qua tin nhắn SMS. Quy trình trao đổi khóa bí mật bằng tin nhắn SMS bao gồm các bước như sau:

- Bước 1: A tạo khóa bí mật và các tham số liên quan đến thuật toán để mã hóa và giải mã tin nhắn.
- Bước 2: A sử dụng khóa công cộng của B để mã hóa khóa và các tham số liên quan.
- Bước 3: A tạo tin nhắn SMS gồm các thông tin đã mã hóa và gởi sang cho B.
- Bước 4: B nhận được tin nhắn chứa thông tin khóa bí mật
- Bước 5: B dùng khóa riêng (Private Key) của mình để giải mã tin nhắn SMS và lưu trữ thông tin khóa giải mã được xuống cơ sở dữ liệu Contact của B.

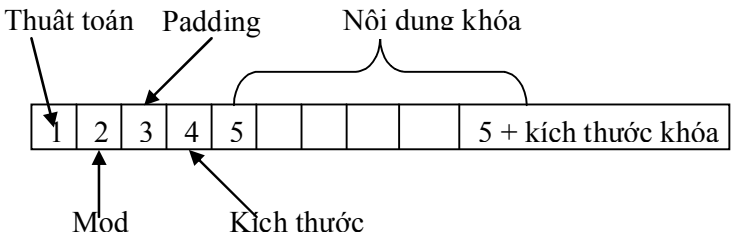


Hình 5.9: Quy trình trao đổi khóa

- **Cấu trúc gói tin trao đổi khóa**

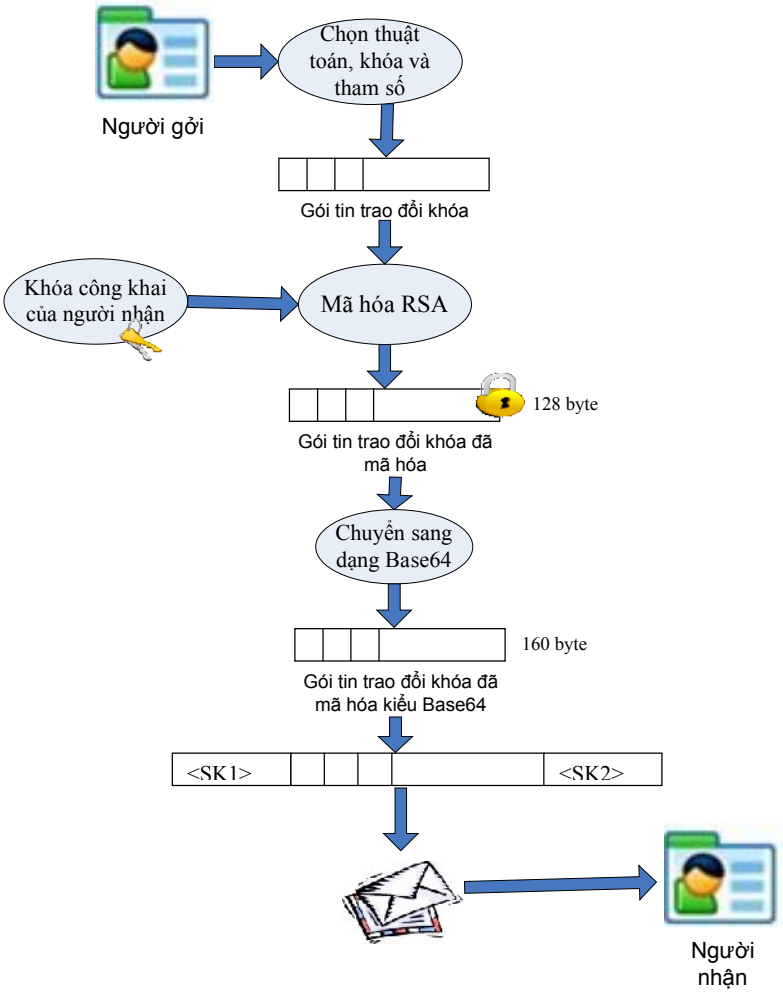
Gói tin trao đổi khóa có kích thước 128 byte. Chứa các thông tin.

- Byte thứ 0: thuật toán được sử dụng.
- Byte thứ 1: kiểu mod được sử dụng trong thuật toán
- Byte thứ 2 kiểu Padding được sử dụng trong thuật toán.
- Byte thứ 4: kích thước khóa được sử dụng
- Byte thứ 5 đến byte 4+kích thước khóa : nội dung khóa
- Các byte còn lại : giá trị rác.



Hình 5.10: Cấu trúc gói tin trao đổi khóa

- Các bước tạo tin nhắn chứa cấu trúc gói tin trao đổi khóa như sau:
  - Bước 1: tạo cấu trúc gói tin trao đổi khóa (hình 5.10) từ các giá trị được người dùng chọn.
  - Bước 2: sử dụng khóa công khai của người nhận để mã hóa gói tin trao đổi khóa vừa tạo ở bước 1.
  - Bước 3: lấy kết quả nhận được ở bước 2, chuyển sang dạng Base64 để có thể gởi được qua mạng **GSM.**
  - Bước 4: tạo tin nhắn với nội dung là kết quả có được ở bước 3. Chèn vào đầu nội dung tin nhắn chuỗi "**<SK1>**" và cuối nội dung tin nhắn chuỗi **"<SK2>"** để đánh dâu đây là tin nhắn trao đổi khóa.
  - Bước 5: gởi tin nhắn vừa tạo đến cho người nhận.



Hình 5.11: Các bước tạo tin nhắn chứa gói tin trao đổi khóa.

- Nhận xét:
  - Thuật toán mã hóa gói tin được sử dụng là RSA với kích thước khóa 1024 bit (128 byte), nên gói tin mã hóa nhận được sau bước 2 có kích thước 128 byte.
  - Sau khi chuyển qua kiểu base64, kết quả gói tin trao đổi khóa nhận được sẽ có kích thước 160 byte. Như vậy, kích thước này lơn hơn chiều dài quy định của tin nhắn. Nên để gởi được gói tin này cần phải dùng đến hai tin nhắn.

5.2.6. Quy trình gởi và nhận tin nhắn

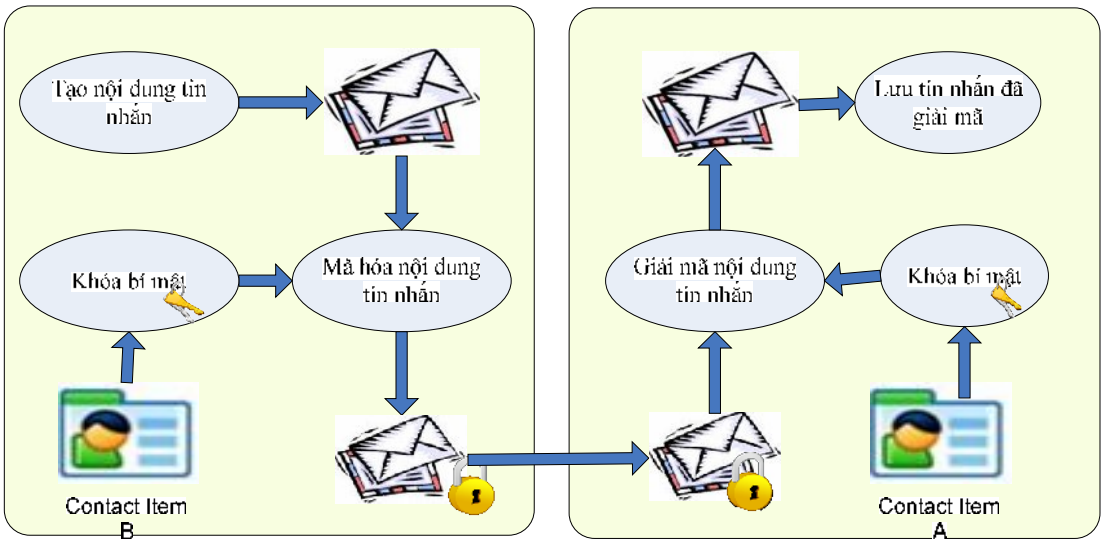
A muốn gởi tin nhắn đã mã hóa cho **B** với khóa và các tham số liên quan đã được trao đổi trước đó. Quy trình này gồm các bước chi tiết sau:

Giai đoạn gởi của A.

- Bước 1: A tạo nội dung tin nhắn để gởi cho B.
- Bước 2: A lấy thông tin khóa đã được trao đổi trước đó trong cơ sỡ dữ liệu Contact để mã hóa nội dung tin nhắn.
- Bước 3: Thực hiện mã hóa nội dung tin nhắn với thông tin khóa tìm được.
- Bước 4: Gởi tin nhắn cho B.

Giai đoạn nhận của B.

- Bước 1: B nhận nội dung tin nhắn vừa được gởi đến từ A.
- Bước 2: Dựa vào thông tin số điện thoại của tin nhắn vừa gởi đến, lấy thông tin khóa bí mật từ cơ sở dữ liệu Contact.
- Bước 3: Sử dụng thông tin khóa vừa tìm được để giải mã nội dung tin nhắn.
- Bước 4: lưu trữ nội dung tin nhắn vừa giải mã và thông tin khóa xuống hệ thống lưu trữ.



Hình 5.12: Quy trình gởi và nhận tin nhắn được mã hóa bằng thuật toán đối xứng.

5.3. Kết luận.

Trong chương này chúng em đã trình bày các quy trình bảo mật mà nhóm đã xây dựng nhằm thiết lập các cơ chế bảo mật cần thiết cho hệ thống thông tin người dùng trên hệ điều hành Andoird. Các quy trình đã trình bày trong chương này đã được đã được hiện thực bằng bộ ứng dụng mà chúng em sẽ trình bày trong chương 6 và 8.

Chương 6

Giới thiệu ứng dụng

✍ *Nội dung của chương 6 giới thiệu về bộ ứng dụng mà nhóm phát triển dựa trên các quy trình đã xây dựng trong chương 5.*

6.1. Giới thiệu bộ ứng dụng

Bộ ứng dụng được xây dựng dựa trên quy trình bảo mật ở chương 5 có những đặc điểm nổi bật sau:

- Bảo mật hệ thống thông tin người dùng bao gồm tin nhắn và contact
- Hỗ trợ đầy đủ các tính năng cần thiết của một chương trình quản lý contact và SMS thông dụng.
- Giao diện chức năng bắt mắt.
- Đáp ứng nhu cầu nhanh chóng.
- Hoạt động ổn định.
- Cung cấp những tính năng giúp người dùng thiết lập và quản lý cấu hình một cách dễ dàng, cho phép người sử dụng lựa chọn thuật toán và những thông tin của riêng mình.

Bộ ứng dụng gồm các chương trình sau:

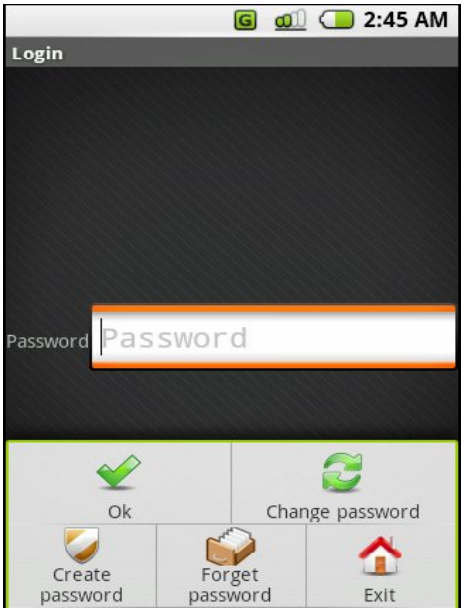
- GPContact: Dùng quản lý thông tin contact, hỗ trợ các tính năng mã hóa bảo mật và một số tính năng khác như: chạy nền ứng dụng, tạo một bộ dữ liệu cho riêng chương trình.
- GPSms: Dùng quản lý thông tin SMS, hỗ trợ thêm các tính năng mã hóa

6.2. Các chương trình trong bộ ứng dụng

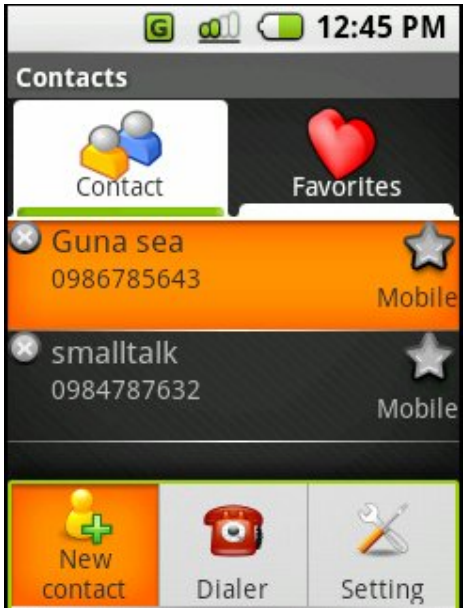
6.2.1 Ứng dụng GPContact

Đây là chương trình quản lí contact hỗ trợ các tính năng: quản lí và bảo mật các thông tin contact trên ứng dụng, quản lí cấu hình để thực thi chương trình. Bao gồm các chức năng chính sau:

Chương trình Contact



Hình 6.1: Màn hình đang nhập ứng dụng



Hình 6.2: Màn hình quản lý danh sách contact

Trước khi sử dụng ứng dụng, ngươi dùng phải đăng nhập (Hình 6.1). Ngoài ra, ứng dụng còn cho phép thay đổi mạt khẩu hay phục hồi lại mật khẩu.

- **Các chức năng của màn hình chính (hình 6.2)**
  - Quản lý danh sách Contact.
  - Xem chi tiết Contact.
  - Tạo một Contact mới.
  - Gọi điện thoại.
  - Cấu hình ứng dụng.
  - Mở thông tin chi tiết một Contact.
  - Quản lý danh sách Contact ưa thích.

- **Chức năng thêm mới Contact**

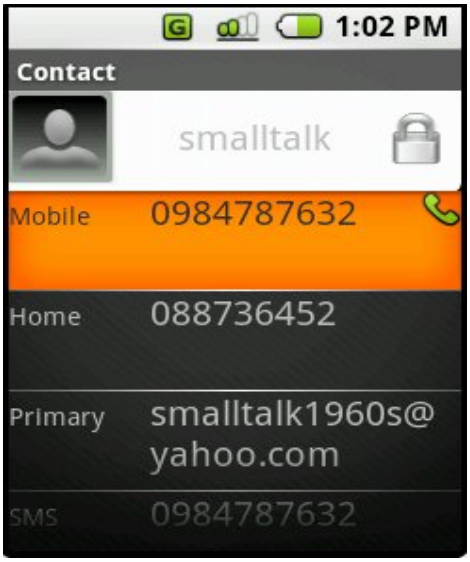
Chức năng này cho phép người dùng tạo mới Contact Item với các thông tin:

  - Số điện thoại
  - Địa chỉ Email
  - Địa chỉ

Hình 6.3: Màn hình thêm thông tin contact

- **Chức năng quản lý chi tiết contact**

- **Edit**: hiệu chỉnh contact
- **Delete**: xóa contact hiện hành
- **Send Secret Key**: gửi khóa bí mật tới contact
- **Search Public Key**: tìm khóa công cộng của contact item
- **Set main phone**: đặt số điện thoại chính
- **Set main email**: đặt email chính
- **Encrypt/Decrypt**: mã hóa và giải mã contact



Hình 6.4: Màn hình xem chi tiết contact

- **Chức năng quản lí cấu hình ứng dụng.**

Quản lí các thông số cấu hình ứng dụng. Gồm các thông tin sau:

- **Algorithm**: thuật toán dùng để mã hóa.
- **Encrypt when inserting**: có thực hiện mã hóa khi thêm contact hay không.
- **Decrypt when logining**: có giải mã khi đăng nhập hay không
- **Timeout**: khoảng thời gian sẽ tự động thoát ứng dụng khi không dùng tới.

Hình 6.5: Màn hình quản lý cấu hình ứng dụng contact

6.2.2 Ứng dụng GPSms

Chương trình quản lý và bảo mật tin nhắn SMS trên điện thoại di động. Một số chức năng chính của ứng dụng như sau: quản lí tin nhắn SMS (soạn thảo, copy, di chuyển và xóa tin nhắn), bảo mật tin nhắn (mã hóa, giải mã, mã hóa tin nhắn khi gửi), quản lý thư mục do người dùng tạo ra, quản lý các tin nhắn mẫu.

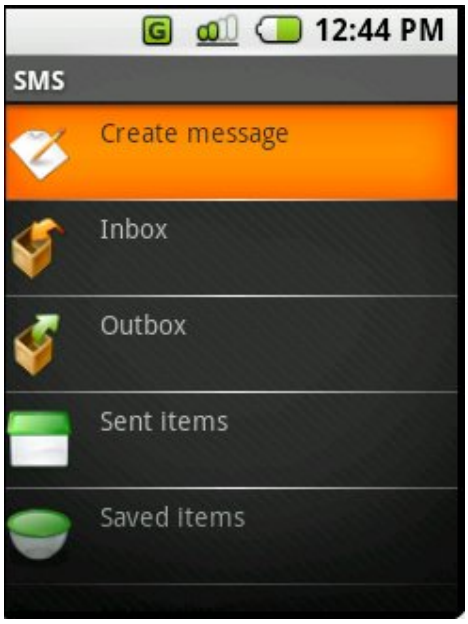
Trước khi sử dụng, người dùng cũng phải đăng nhập vào ứng dụng (tương tự như ứng dụng Contact), cho phép thay đổi mật khẩu hay phục hồi lại mạt khẩu.

Một số màn hình chức năng chính của ứng dụng như sau

- Màn hình giao diện chính

Màn hình bao gồm các chức năng sau:

- **Create message**: tạo tin nhắn mới
- **Inbox**: quản lí danh sách tin nhắn tới
- **Outbox**: các danh sách tin nhắn hỏng khi gửi
- **Sent items**: các danh sách tin đã gửi
- **Saved items**: các danh sách tin đã lưu
- **My folders**: quản lí các folder
- **My templates:** quản lí các mẫu tin nhắn
- **Delete messages:** quản lí việc xóa tin nhắn
- **Message Settings:** quản lí cấu hình ứng dụng
- **Help:** màn hình giúp đỡ
- **Exit:** thoát khỏi ứng dụng

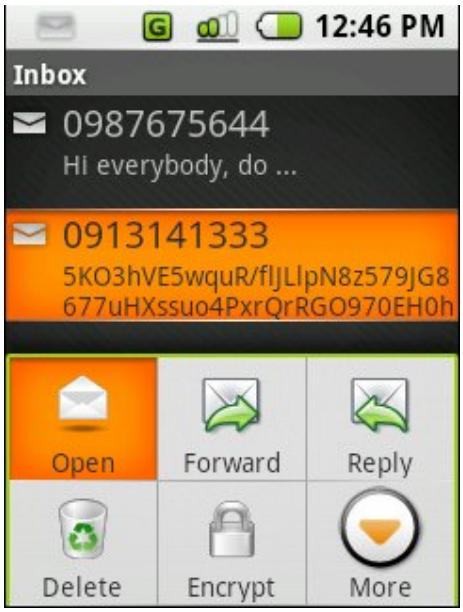


Hình 6.6: Màn hình chính ứng dụng sms

- Chức năng quản lí danh sách tin nhắn

Các màn hình quản lí tin nhắn liệt kê các tin theo dạng danh sách. Gồm các chức năng chính:

- Xem chi tiết một tin nhắn.
- Xóa tin nhắn đang chọn
- Mã hóa và giải mã tin nhắn
- Rely hay Forward tin nhắn đang chọn
- Di chuyển tin nhắn sang thư mục khác



Hình 6.7: Màn hình quản lý danh sách tin nhắn

- Chức năng cấu hình ứng dụng:

Gồm các thông tin cấu hình sau:

- **Algorithm**: thuật toán dùng để mã hóa.

- **Encrypt when inserting:** có mã hóa khi thêm contact hay không

- **Decrypt when logining':** có giải mã khi đăng nhập hay không

- **Timeout:** khoảng thời gian sẽ tự động thoát ứng dụng khi không dùng tới.

- **encrypt incoming message** : tự động mã hóa tin nhắn đến

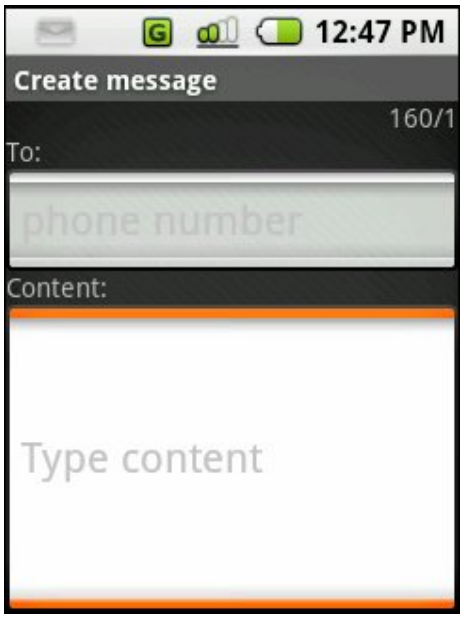
- **ask key when send sms :** yêu cầu khóa để mã hóa khi gởi tin nhắn.

Hình 6.8: Màn hình cấu hình ứng dụng sms

- Chức năng soạn thảo tin nhắn

Hỗ trợ gửi tin nhiều người. Bao gồm các chức năng chính sau:

- Gửi tin nhắn
- Tùy chọn khi gửi tin nhắn dưới dạng mã hóa
- Chèn số điện thoại khi gởi tin nhắn
- Lưu lại tin nhắn đang soạn thảo
- Thêm nội dung tin nhắn từ các mẫu (template) có sẵn
- Lưu tin nhắn dưới dạng template.



Hình 6.9: Màn hình viết tin nhắn

- Chức quản lí thư mục

Quản lí thư mục ứng dụng (Inbox, Outbox, Sentitem, Saveditem) và thư mục do người dùng tạo ra. Gồm 4 chức năng

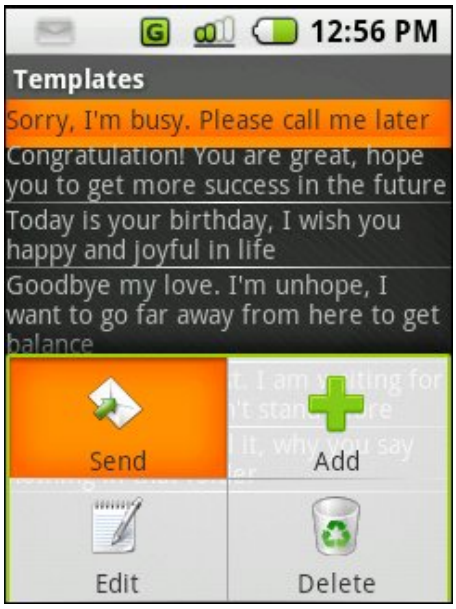
- Thêm một thư mục mới (Add)
- Mở thư mục đang chọn
- Hiệu chỉnh tên thư
- Xóa thư mục đang chọn.

Hình 6.10: Màn hình quản lý thư mục

- Chức năng quản lý template

Liệt kê các mẫu tin theo dạng danh sách. Bao gồm các chức năng

- Dùng mẫu tin để gửi (Send)
- Thêm mẫu tin mới (Add)
- Hiệu chỉnh mẫu tin (Edit)
- Xóa mẫu tin hiện đang chọn (Delete)



Hình 6.11: Màn hình quản lý template

6.3. Kết luận

Trong chương 6, chúng em đã giới thiệu về bộ ứng dụng dùng để bảo mật cho các hệ thống thông tin người dùng bao gồm Contact, Sms. Đây là nội dung tóm tắt giúp người dùng có cái nhìn tổng thể về ứng dụng. Chi tiết về chương trình sẽ được trình bày bày trong **Chương 7** (Ứng dụng Contact) và **Chương 8** (ứng dụng SMS) của báo cáo này.

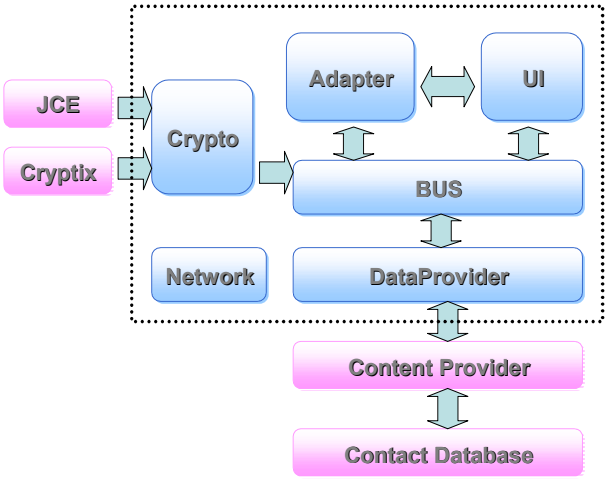
Chương 7

Kiến trúc hệ thống & Chức năng ứng dụng GPContact

Nội dung của chương 7 mô tả kiến trúc ứng dụng GPContact xây dựng theo quy trình đã xây dựng ở Chương 5 và hướng dẫn sử dụng ứng dụng.

7.1. Kiến trúc hệ thống

7.1.1. Kiến trúc ứng dụng



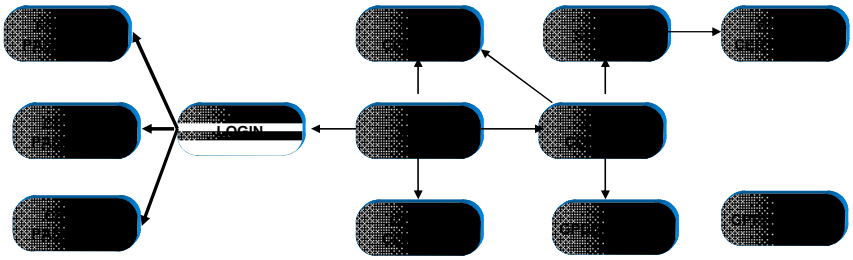
Hình 7.1 Sơ đồ tổng quan kiến trúc của ứng dụng

Thành phần	Ý nghĩa
UI	Quản lý giao diện người dùng
BUS	Quản lý nghiệp vụ, xử lý dữ liệu.
DATAPROVIDER	Quản lý kết nối cơ sở dữ liệu, sử dụng Content Provider để truy cập Cơ sở dữ liệu Contact.
CRYPTO	Quản lý bảo mật, hỗ trợ một số thuật toán thông dụng, sử dụng thư viện mã hóa JCE và Cryptix-JCE
OBJECT	Các đối tượng đã được đóng gói sẵn, hỗ trợ nhanh quá trình trao đổi và xử lý thông tin.
ADAPTER	Quản lý hiển thị giao diện, qui định cách bố trí, các bố cục giao diện
NETWORK	Thành phần giả lập gởi tin nhắn. theo nghi thức mạng
JCE và Cryptyx	Thư viện mã hóa, xem phụ lục A.

Bảng 7.1 Các thành phần trong ứng dụng

7.1.2. Sơ đồ chi tiết các thành phần

- **Thành phần quản lí giao diện (UI)**

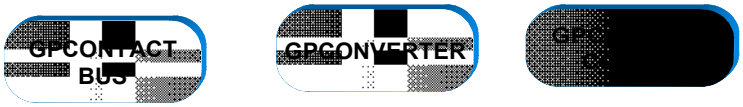


Hình 7.2 Các lớp trong thành phần quản lí giao diện người dùng.

Lớp	Loại	Diễn giải
Login	Hiển thị	Lớp hiển thị màn hình đăng nhập
Main	Hiển thị	Lớp hển thị màn hình chính của chương trình quản lý danh sách Contact
ChangePassword	Hiển thị	Lớp hiển thị màn hình thay đổi mật khẩu
CreatePassword	Hiển thị	Lớp hiển thị màn hình tạo mật khẩu
ForgetPassword	HIển thị	Lớp hiển thị màn hình khôi phục mật khẩu
GPConfigContact	Hiển thị	Lớp hiển thị màn hình cấu hình ứng dụng
EditContact	Hiển thị	Lớp hiển thị màn hình hiệu chỉnh contact, thêm một Contact Item mới.
ViewContact	Hiển thị	Lớp hiển thị màn hình xem chi tiết contact
Search	Hiển thị	Lớp hiển thị màn hình tìm kiếm chứng nhận điện tử
DetailCertificate	Hiển thị	Lớp hiển thị màn hình xem chi tiết chứng nhận điện tử
GPProgressBar	Hiển thị	Màn hình thanh trạng thái
GPDialogKey	Đối tượng	Màn hình xác lập khóa tra đối.

Bảng 7.2 Các lớp trong thành phần quản lý giao diện.

- **Thành phần quản lý nghiệp vụ (BUS)**



Hình 7.3 Các lớp trong thành phần nghiệp vụ.

Lớp	Loại	Diễn giải
GPContactsBus	Xử lý	Lớp xử lí nghiệp vụ contact, điều phối dữ liệu giữa tầng giao diện và tầng truy xuất dữ liệu.
GPConverter	Xử lý	Lới xử lý chuyển đổi dữ liệu sang kiểu Base64.
GPContactConfig	Xử lý	Lớp xử lý thông tin cấu hình hệ thống

Bảng 7.3 Thành phần quản lý nghiệp vụ.

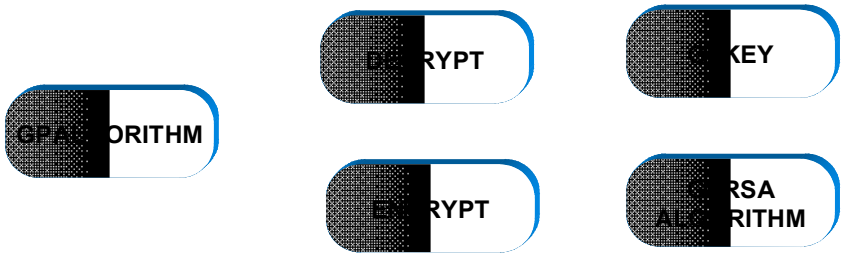
- **Thành phần quản lý kết nối dữ liệu (DataProvider)**

Hình 7.4 Các lớp trong thành phần quản lý kết nối dữ liệu

Lớp	Loại	Diễn giải
GPContactsProvider	Xử lý	Lớp xử lí kết nối, truy xuất dữ liệu.
GPXml	Xử lý	Đối tượng xử lý đọc dữ liệu dữ liệu XML được trả về từ CA..

Bảng 7.4 Các lớp trong thành phần quản lý kết nối dữ liệu

- Thành phần bảo mật (CRYPTO)

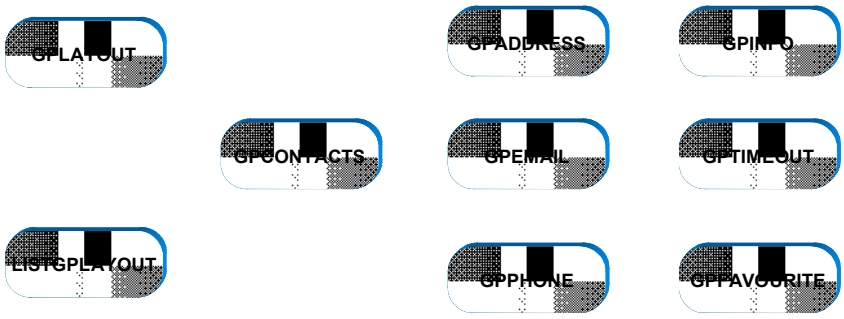


Hình 7.5 Các lớp trong thành phần Crypto

Lớp	Loại	Diễn giải
Decrypt	Xử lý	Lớp xử lý giải mã
Encrypt	Xử lý	Lớp xử lý mã hóa
GPAlgorithm	Xử lý	Lớp xử lý thuật toán, quản lý các tham số để mã hóa và giải mã dữ liệu.
GPKey	Xử lý	Lớp xử lý khóa bảo mật, phát sinh khóa.
GPRsaAlgorithm	Xử lý	Lớp xử lý thuật toán RSA, mã hóa và giải mã dữ liệu.

Bảng 7.5 Các lớp trong thành phần bảo mật.

- **Thành phần đối tượng (OBJECT)**

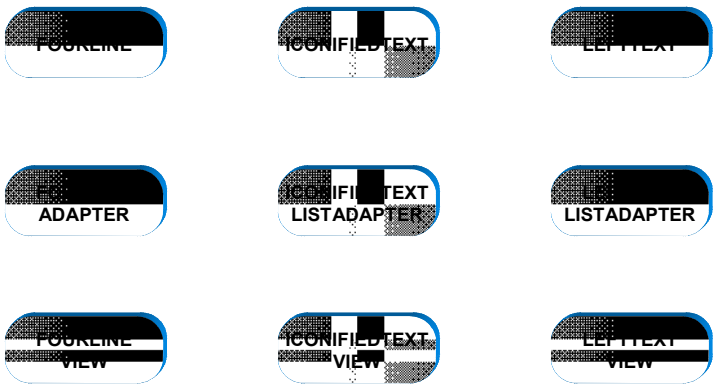


Hình 7.6 Các lớp trong thành phần Object

Lớp	Loại	Diễn giải
GPAddress	Đối tượng	Đối tượng xử lý address
GPContacts	Đối tượng	Đối tượng xứ lý thông tin contact
GPEmail	Đối tượng	Đối tượng xử lý thông tin email
GPFavourite	Đối tượng	Đối tượng favourite
GPInfo	Đối tượng	Đối tượng info
GPLayout	Đối tượng	Đối tượng layout.
GPPhone	Đối tượng	Đối tượng phone
GPTimeOut	Đối tượng	Đối tượng timeout
ListGPLayout	Đối tượng	Đối tượng layout

Bảng 7.6 Các lớp trong thành phần quản lý đối tượng

- Thành phần quản lí hiển thị dữ liệu (ADAPTER)



Hình 7.7 Các lớp trong thành phần điều phối dữ liệu.

Lớp	Loại	Diễn giải
FourLine	Xử lý	Đối tượng tự định nghĩa
FourLineAdapter	Xử lý	Bộ điều phối kết nối dữ liệu
FourLineView	Hiển thị	Bố cục của đối tượng
IconifiedText	Xử lý	Đối tượng tự định nghĩa
IconifiedTextListAdapter	Xử lý	Bộ điều phối kết nối dữ liệu
IconifiedTextView	Hiển thị	Bố cục của đối tượng
LeftText	Xử lý	Đối tượng tự định nghĩa
LeftTextListAdapter	Xử lý	Bộ điều phối kết nối dữ liệu
LeftTextView	Hiển thị	Bố cục của đối tượng dữ liệu hiển thị trên màn hình.

Bảng 7.7 Các lớp trong thành phần điều phối dữ liệu

- Thành phần quản lý mạng (NETWORK)

Hình 7.8 Các lớp trong gói quản lý giả lập gởi tin nhắn.

Lớp	Diễn giải
SmsNetwork	Lớp xử lý giả lập gởi tin nhắn qua mạng

Bảng 7.8 Các lớp trong gói quản lý giả lập gởi tin nhắn.

7.2. Các chức năng chính

7.2.1. Đăng nhập hệ thống

Trước khi sử dụng ứng dụng, người dùng phải đăng nhập vào hệ thống.Ngoài ra người dùng có thể thay đổi mật khẩu đăng nhập, tạo câu hỏi bí mật để có thể phục hồi lại mật khẩu trong trường hợp quên mạt khẩu đăng nhập.

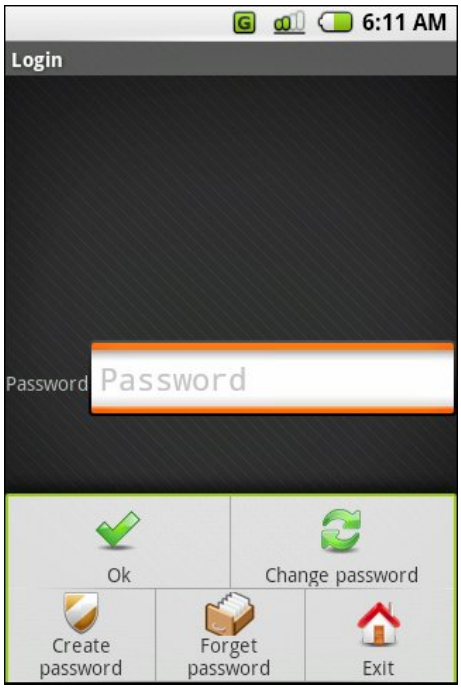
Lưu ý, trong lần đầu tiên chạy chương trình. Người dùng phải đăng nhập với mật khẩu mặc định là **"24091711",** sau đó chọn chức năng thay đổi mật khẩu để thay đổi mật khẩu.

Nhập mật khẩu đăng nhập sau đó chọn **OK** để đăng nhập vào hệ thống.

Để thay đổi mật khẩu đăng nhập chọn chức năng "**Change Password".**

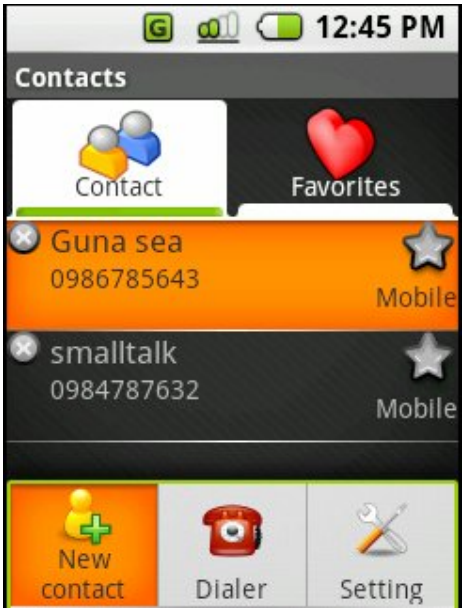
Để hỗ trợ lấy lại mật khẩu trong trường hợp quên mật khẩu đăng nhập chọn chức năng **"Create Password"** để tạo cấu hỏi bí mật.

Và trong trường hợp quên mật khẩu đăng nhập, nếu người dung đã tạo câu hỏi bí mật trước đó thì chọn chức năng **"Forget Password"** để lấy lại mật khẩu đăng nhập.



Hình 7.9: màn hình đang nhập ứng dụng

7.2.2. Thao tác trên màn hình chính



Hình 7.10: Màn hình quản lý danh sách contact

Chức năng	Ý nghĩa
New contact	Chức năng tạo mới Contact Item
Dialer	Gọi điện thoại. Gọi tới chương trình gọi Dialer trong Android  Hình 7.11: Màn hình quay số điện thoại
Setting	Gọi thực hiện chức năng quản lý cấu hình ứng dụng

Bảng 7.9 Chúc năng màn hình chính

7.2.3. Thêm mới Contact

Hình 7.12: Màn hình thêm mới contact Hình 7.13: Màn hình thêm thông tin mới

Chức năng	Ý nghĩa
Done	Chấp nhận tạo mới Contact Item
Revert	Bỏ qua, không lưu thông tin vừa mới thao tác.
Addmore	Thêm một số thông tin phụ như số điện thoại, địa chỉ email, hoặc địa chỉ.

Bảng 7.10: Chức năng của màn hình thêm Contact

7.2.4. Chức năng quản lý cấu hình

Hình 7.14: Màn hình quản lý cấu hình ứng dụng contact

Chức năng	Ý nghĩa
OK	Chấp nhận các thông , lưu xuống file cấu hình trong hệ thống. Thông số cấu hình sẽ có hiệu lực ngay lập tức, không cần khởi động lại ứng dụng.
Revert	Hủy các thông số vừa hiệu chỉnh

Bảng 7.11: Chức năng màn hình quản lý cấu hình GPContact

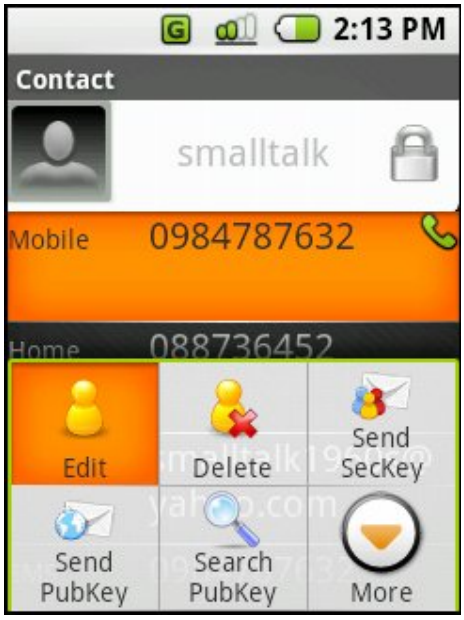
Người dùng có thể cấu hình các thông số sau:

- **Algorithm** : lựa chọn thuật toán để mã hóa
- **Encrypt when inserting** : tự động mã hóa ngay khi thêm mới một Contact Item
- **Decrypt when logining** : tự động giải mã thông tin Contact khi đăng nhập vào ứng dụng.

7.2.5. Chức năng xem chi tiết Contact

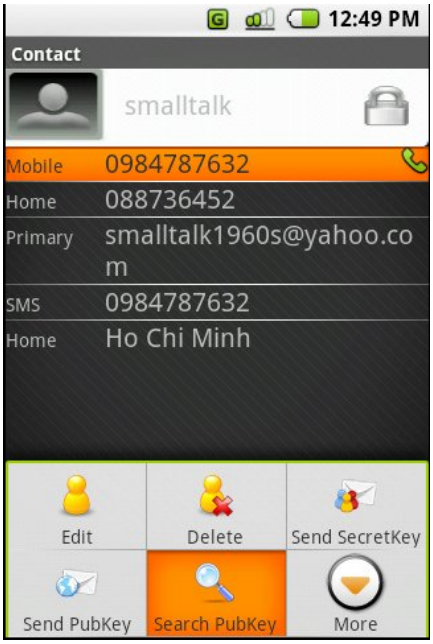
Chức năng	Ý nghĩa
Edit contact	Hiểu chỉnh thông tin Contact Item đang xem chi tiết
Delete contact	Xóa contact hiện hành
Set main phone	Thiết lập số điện thoại chính
Set main email	Thiết lập email chính
Encrypt	Mã hóa thông tin nếu chưa được mã hóa
Decrypt	Giải mã thông tin nếu đã được mã hóa
Send SecretKey	Gửi thông tin khóa bí mật tới contact (xem phần 7.2.7 Trao đổi khóa bí mật).
Send PublicKey	Gửi thông tin khóa công tới contact
Search PublicKey	Tìm kiếm thông tin khóa công ứng với contact item đang xem chi tiết. Chức năng này sẽ kết nối với Web Service (Xem phần 7.2.6 tra cứu khóa công khai).

Bảng 7.12: Danh sách chức năng quản lý chi tiết Contact.



Hình 7.15: Màn hình quản lý chi tiết contact

7.2.6. Chức năng tra cứu khóa công khai



Hình 7.16 Màn hình xem chi tiết contact

Hình 7.17 Màn hình tìm chứng nhận

Trong màn hình ViewContact chọn chức năng Search PublicKey, để tra cứu khóa công khai trên Web Service. Khi chọn chức năng Search PublicKey, ứng dụng sẽ hiển thị màn hình (7.17) cho phép người dùng tùy chọn các chức năng tra cứu theo tên, địa chỉ email hày số điện thoại. Nếu tìm thầy, ứng dụng sẽ trả về danh sách các người dùng có thông tin giống với thông tin tìm kiếm

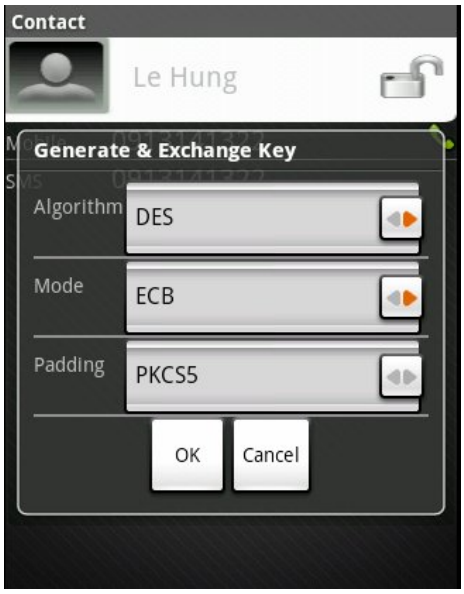
Khi chọn xem chi tiết item nào, thì ứng dụng sẽ hiển thị màn hình sao cho người dùng xem chi tiết thông tin Public Key của thông tin được chọn

Hình 7.18 Màn hình chi tiết chứng nhận

7.2.7. Trao đổi khóa bí mật

Để sử dụng chức năng trao đổi khóa, trong thông tin Contact Item phải có khóa công khai của người liên lạc cần trao đổi khóa. Nếu chọn chức năng này mà chưa có khóa công khai thì chương trình sẽ báo lỗi.

Nếu trong Contact Item đã có khóa công khai thì chương trình sẽ hiển thị màn hình sau cho người dùng chọn các tham số để ứng dụng phát sinh khóa bí mật:



Hình 7.19: Màn hình thiết lập thông số trao đổi khóa.

Người dùng chọn các thông số thích hợp bao gồm thuật toán, kiểu Mode, kiểu Padding. Nếu chấp nhận hết các thông số thì chọn **OK,** chương trinh sẽ phát sinh khóa và tạo tin nhắn chứa thông tin vừa xác lập đến Contact Item đang chọn.

7.3. Kết luận

Trong chương này chúng em đã trình bày tổng quan về kiến trúc ứng dụng Contact, giới thiệu sơ lược về vai trò của các lớp trong ứng dụng. Đồng thời trình bày tóm tắt các chức năng chương trình hỗ trợ người dùng. Trong chương 8, chúng em sẽ tiếp tục giới thiệu về chương trình GPSms trong bộ ứng dụng.

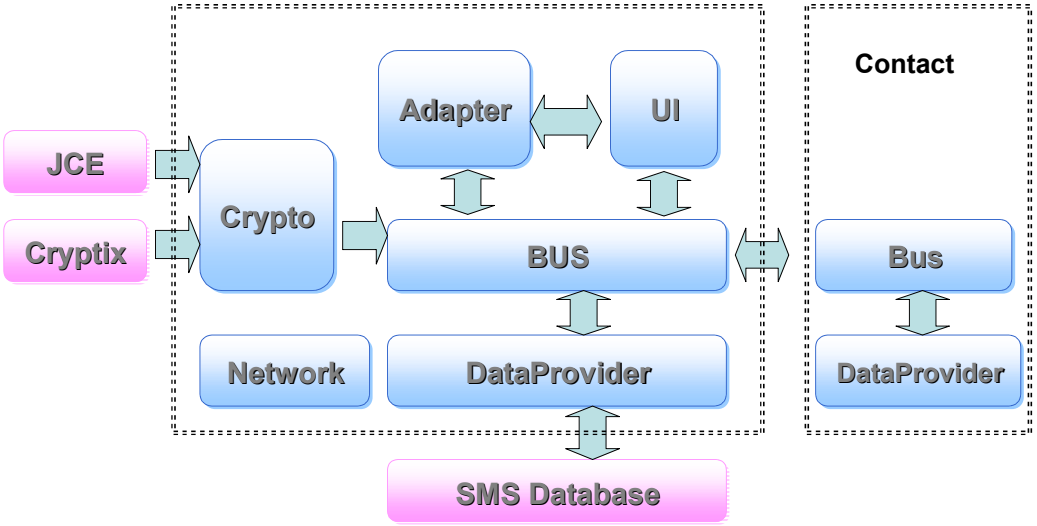
Chương 8

Kiến trúc hệ thống và chức năng ứng dụng SMS

✍ *Nội dung của chương 8 mô tả kiến trúc ứng dụng GPSms xây dựng theo quy trình đã xây dựng ở Chương 5 và hướng dẫn sử dụng các chức năng của ứng dụng.*

8.1. Kiến trúc ứng dụng

8.1.1. Sơ đồ tổng quan các thành phần



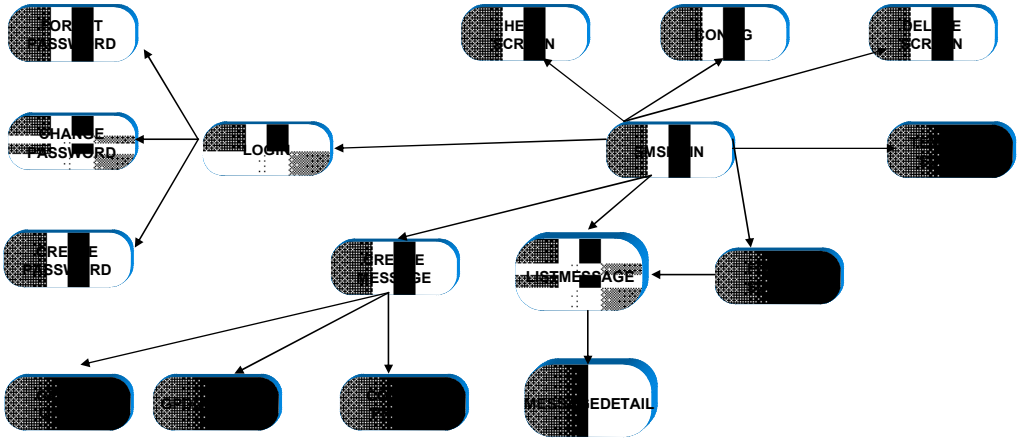
Hình 8.1: Sơ đồ tổng quan các thành phần của ứng dụng.

Thành phần	Ý nghĩa
UI	Quản lý giao diện người dùng
BUS	Quản lý nghiệp vụ, xử lý dữ liệu.
DATAPROVIDER	Quản lý kết nối cơ sở dữ liệu, sử dụng Content Provider để truy cập Cơ sở dữ liệu Contact.
CRYPTO	Quản lý bảo mật, hỗ trợ một số thuật toán thông dụng, sử dụng thư viện mã hóa JCE và Cryptix-JCE. Tương tự giống gói Crypto trong ứng dụng GPContact.
OBJECT	Các đối tượng đã được đóng gói sẵn, hỗ trợ nhanh quá trình trao đổi và xử lý thông tin.
ADAPTER	Quản lý hiển thị giao diện, qui định cách bố trí, các bố cục giao diện
NETWORK	Thành phần giả lập gởi tin nhắn. theo nghi thức mạng
JCE và Cryptyx	Thư viện mã hóa được sử dụng trong ứng dụng, xem phụ lục A.

Bảng 8.1: Danh sách các thành phần trong ứng dụng SMS.

8.1.2. Sơ đồ chi tiết các thành phần

- Thành phần quản lí giao diện (gói UI)

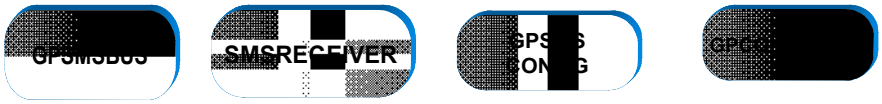


Hình 8.2 Sơ đồ tương tác giữa các gói trong thành phần giao diện

Thành phần	Loại	Diễn giải
LoginScreen	Hiển thị	Lớp hiển thị màn hình đăng nhập
SMSMainScreen	Hiển thị	Lớp chính của chương trình
ChangePasswordScreen	Hiển thị	Lớp hiển thị màn hình thay đổi mật khẩu
CreatePasswordScreen	Hiển thị	Lớp hiển thị màn hình tạo mật khẩu
ForgetPasswordScreen	HIển thị	Lớp hiển thị màn hình khôi phục mật khẩu
MessageSettingScreen	Hiển thị	Lớp hiển thị màn hình cấu hình ứng dụng
AlertScreen	Hiển thị	Lớp hiển thị thông báo khi có tin nhắn đến
ContactPhonesScreen	Hiển thị	Lớp hiển thị số điện thoại contact
ContactScreen	Hiển thị	Lớp hiển thị thông tin contact
CreateMessageScreen	Hiển thị	Lớp hiển thị màn hình soạn tin nhắn
DeleteScreen	Hiển thị	Lớp hiển thị màn hình xóa tin nhắn
FolderScreen	Hiển thị	Lớp hiển thị màn hình quản lý thư mục
HelpScreen	Hiển thị	Lớp hiển thị màn hình giúp đỡ
MessageScreen	Hiển thị	Lớp hiển thị màn hình tin nhắn
MyFolderSscreen	Hiển thị	Lớp hiển thị chi tiết thư mục
OpenScreen	Hiển thị	Lớp hiển thị chi tiết tin nhắn
TemplateScreen	Hiển thị	Lớp hiển thị màn hình quản lí mẫu tin
GPDialogKey	Đối tượng	Đối tượng xử lý hiển thị hộp thoại khóa
GPSendOption	Đối tượng	Đối tượng xử lý hiển thị hộp thoại tùy chọn gủi

Bảng 8.2 Thành phần quản lí giao diện

- Thành phần nghiệp vụ (BUS)

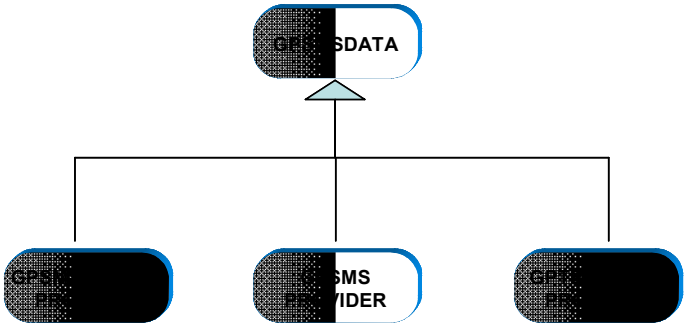


Hình 8.3: Các lớp trong thành phần quản lý nghiệp vụ.

Thành phần	Loại	Diễn giải
GPSmsBus	Xử lý	Lớp xử lý nghiệp vụ Sms
GPConverter	Xử lý	Lớp chịu trách nhiệm chuyển đổi kiểu dữ liệu sang base64
SMSReceiver	Hiển thị	Lớp xử lý khi có tin nhắn đến
GPSmsConfig	Đối tượng	Đối tượng xử lý thông tin cấu hình ứng dụng

Bảng 8.3: Các lớp trong thành phần xử lý nghiệp vụ

- Thành phần quản lý kết nối dữ liệu (DataProvider)

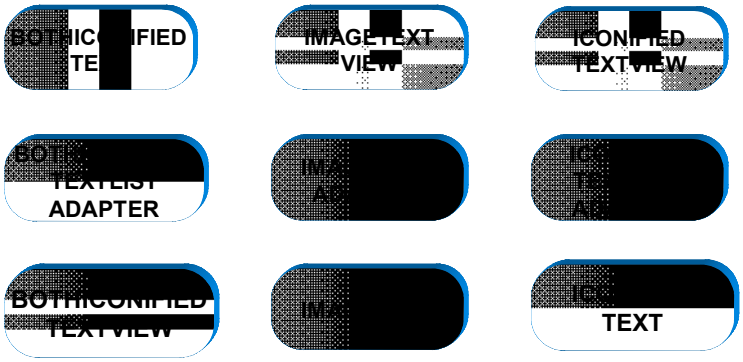


Hình 8.4 Các lớp trong thành phần quản lý kết truy xuất dữ liệu

Thành phần	Loại	Diễn giải
GPSmsData	Xử lý	Lớp xử lý dữ liệu Sms
GPSmsFolderProvider	Xử lý	Lớp tiện ích xử lý dữ liệu thư mục
GPSmsProvider	Xử lý	Lớp tiện ích xử lý dữ liệu Sms
GPSmsTemplateProvider	Xử lý	Lớp tiện ích xử lý dữ liệu mẫu tin

Bảng 8.4 Các lớp trong thành phần kết nối dữ liệu.

- Thành phần điều phối dữ liệu (ADAPTER)

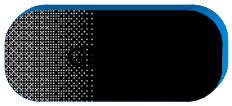


Hình 8.5 Các lớp trong thành phần điều phối dữ liệu

Thành phần	Loại	Diễn giải
BothIconifiedText	Xử lý	Đối tượng tự định nghĩa
BothIconifiedTextListAdapter	Xử lý	Bộ điều phối kết nối dữ liệu
BothIconifiedTextView	Hiển thị	Bố cục của đối tượng
IconifiedText	Xử lý	Đối tượng tự định nghĩa
IconifiedTextListAdapter	Xử lý	Bộ điều phối kết nối dữ liệu
IconifiedTextView	Hiển thị	Bố cục của đối tượng
ImageText	Xử lý	Đối tượng tự định nghĩa
ImageTextAdapter	Xử lý	Bộ điều phối kết nối dữ liệu
ImageTextView	Hiển thị	Bố cục của đối tượng

Bảng 8.5 Chi tiết các lớp trong thành phần điều phối dữ liệu.

- Các lớp đối tượng trong thành phần xứ lý thông tin (Object)



Hình 8.6: Các lớp trong thành phần xử lý thông tin

Thành phần	Loại	Diễn giải
GPSms	Đối tượng	Đối tượng xử lý thông tin SMS

Bảng 8.6 Các lớp trong thành phần xử lý thông tin.

8.2. Các chức năng chính

8.2.1. Đăng nhập hệ thống

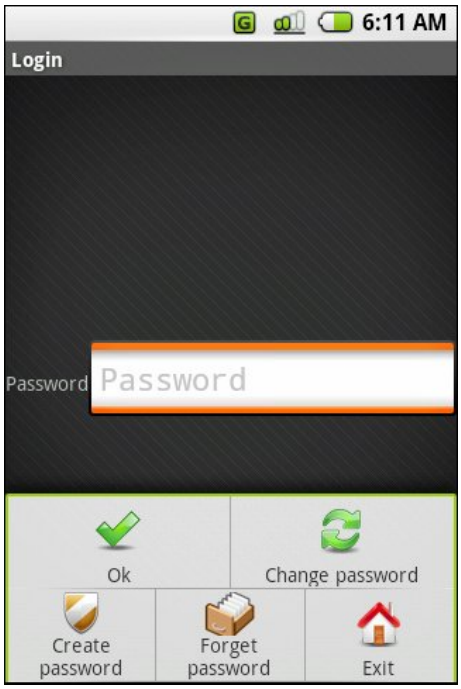
Tương tự như ứng dụng Contact, trước khi sử dụng chương trình người dùng phải đăng nhập vào hệ thống. Người dùng có thể thay đổi mật khẩu đăng nhập, phục hồi mật khẩu.

Nhập mật khẩu đăng nhập sau đó chọn **OK** để đăng nhập vào hệ thống.

Để thay đổi mật khẩu đăng nhập chọn chức năng "**Change Password".**

Để hỗ trợ lấy lại mật khẩu trong trường hợp quên mật khẩu đăng nhập chọn chức năng **"Create Password"** để tạo cấu hỏi bí mật.

Và trong trường hợp quên mật khẩu đăng nhập, nếu người dung đã tạo câu hỏi bí mật trước đó thì chọn chức năng **"Forget Password"** để lấy lại mật khẩu đăng nhập.

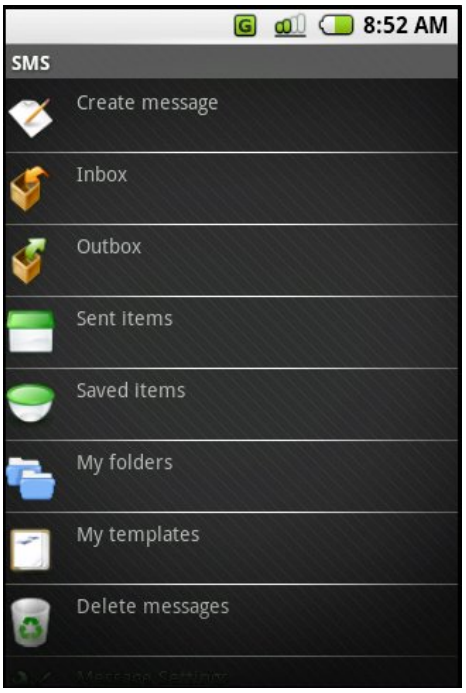


Hình 8.7: màn hình đang nhập ứng dụng

8.2.2. Màn hình chính

Chức năng	Ý nghĩa
Create message	Tạo tin nhắn mới
Inbox	Quản lý các tin vừa nhận
Outbox	Quản lí các tin đã gởi không được
Sent items	Quản lí các tin đã gởi
Saved items	Quản lí các tin đã lưu
My folders	Quản lí danh sách thư mục do người dùng tạo ra
My templates	Quản lí các mẫu tin.
Delete message	Xóa tin nhắn (hỗ trợ xóa các tin trong thư mục hệ thống).
Help	Mở màn hình giúp đỡ
Exit	Thoát khỏi chương trình

Bảng 8.7 Danh sách chức năng màn hình chính



Hình 8.8: Màn hình chính ứng dụng sms

8.2.3. Chức năng tạo tin nhắn

Người dùng có thể gởi tin nhắn cho nhiều người, hỗ trợ gởi tin nhắn theo dạng mã hóa. Ngoài ra, có thể sử dụng các tin nhắn có sẵn (template) hoặc lưu lại tin nhắn.

Hình 8.9 Màn hình tạo tin nhắn

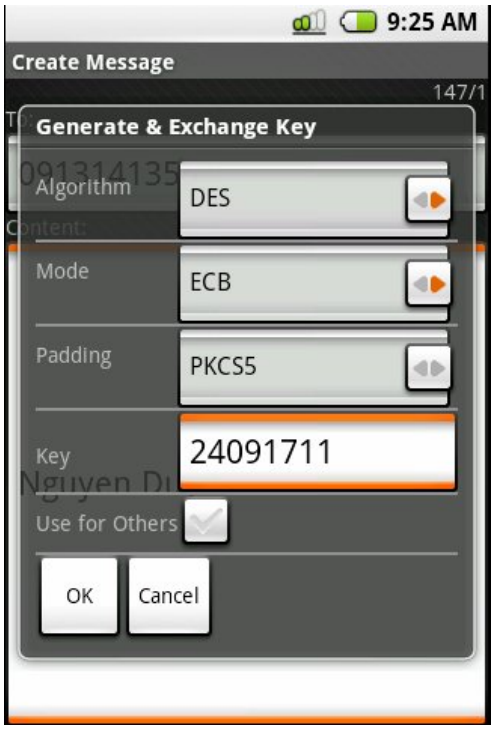
Chức năng	Ý nghĩa
Send message	Gửi tin nhắn đi
Send option	Tùy chọn tham số khi gửi tin nhắn bình thường hay mã hóa trước khi gởi
Clear text	Xóa trắng màn hình
Insert contact	Thêm số điện thoại vào danh sách gởi tin nhắn.
Use template	Dùng các mẫu tin có sẵn để gởi tin nhắn.
Save message	Lưu tin nhắn
Exit	Thoát khỏi màn hình tạo tin nhắn

Bảng 8.8 Danh sách chức năng soạn tin nhắn

8.2.4. Gởi tin nhắn theo dạng mã hóa

Để gởi tin nhắn theo dạng mã hóa, trong chọn **Sending Option** trong màn hình tạo tin nhắn, chọn **Encrypt**

Hình 8.10: Hộp thoại tùy chọn gởi tin theo dạng mã hóa



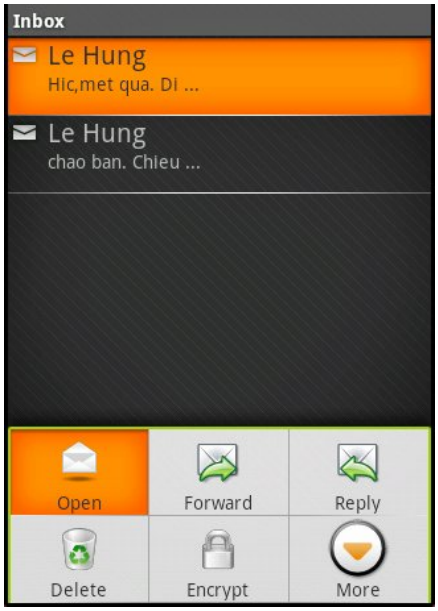
Hình 8.11: Màn hình chọn các tham số để mã hóa tin nhắn

Nếu số điện thoại của người nhận có trong Cơ sở dữ liệu Contact và có khóa trao đổi với người nhận thì ứng dụng sẽ tiến hành mã hóa nội dung tin nhắn và gởi đến cho nguờii nhận

Ngược lại, nếu không tồn tại số điện thoại của người nhận, hoặc không tồn tại một khóa bí mật với người nhận trong cơ sở dữ liệu Contact thì ứng dụng sẽ hiển thị hộp thoại cho người dùng chọn các tham số để mã hóa tin nhắn:

Người dùng có thể chọn các tham số như thuật toán (Algorithm), kiểu Mode, kiểu Padding và Khóa (Key) để mã hóa tin nhắn. Người dùng có thể chọn **(Use for Others)** để chọn các tham số này cho lần gởi sau.

8.2.5. Thao tác trên các màn hình quản lí danh sách tin nhắn (Inbox, SavedItem, SentItem, OutBox) và xem chi tiết tin nhắn



Hình 8.12 Màn hình quản lý danh sách tin nhắn

Hình 8.13 Màn hình xem chi tiết tin nhắn

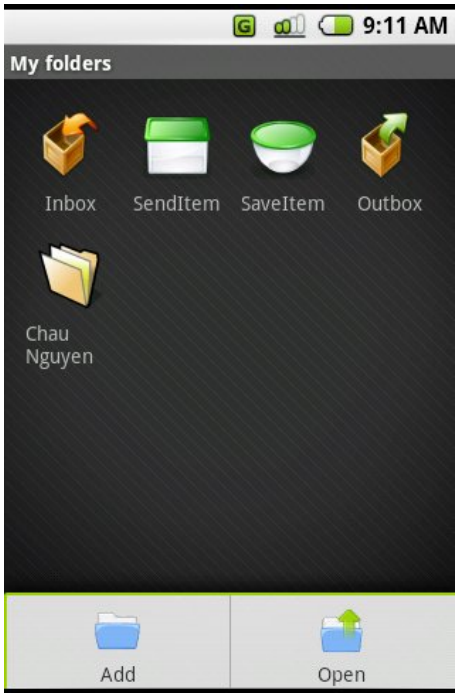
Chức năng	Ý nghĩa
Delete	Xóa tin nhắn đang chọn.
Open	Mở tin nhắn đang chọn
Encrypt	Mã hóa tin nhắn đang chọn
Decrypt	Giải mã tin nhắn đang chọn
Send	Gửi tin nhắn đi

Chức năng	Ý nghĩa
Edit	Hiệu chỉnh nội dung tin nhắn
Move	Di chuyển tin nhắn tới các thư mục khác
Forward	Gởi tin nhắn này đến một số điện thoại khác
Reply	Gởi lại tin nhắn cho người gởi.
Use Detail	Sử dụng chi tiết tin nhắn
Exit	Thoát khỏi màn hình.

Bảng 8.9: danh sách chức năng của ứng chức năng quản lý danh sách tin nhắn

Ngoài ra, người dùng có thể chọn chức năng Use Detail để chọn lưu số điện thoại. Nếu tin nhắn chứa khóa trao đổi, thì chọn chức năng Use Detail để lấy nội dung tin nhắn và lưu xuống cơ sở dữ liệu Contact.

8.2.6. Quản lý thư mục



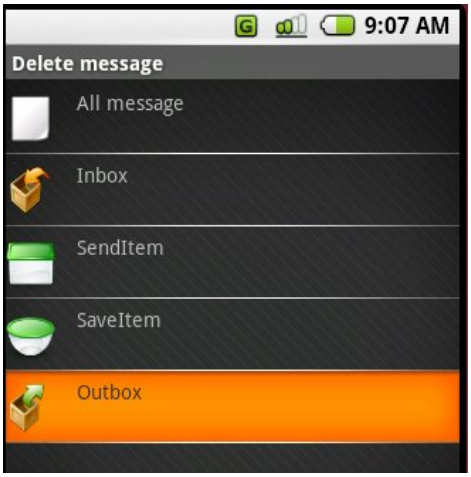
Hình 8.14 Màn hình quản lý thư mục

Chức năng	Ý nghĩa
Add	Thêm thư mục mới
Open	Mở thư mục
Rename	Hiệu chỉnh tên thư mục. Chức năng này chỉ cho phép hiệu chỉnh tên thư mục do người dùng tạo ra.
Delete	Xóa thư mục đang chọn. Chức năng này chỉ cho phép người dùng xóa thư mục do người dùng tạo ra.

Bảng 8.10 Danh sách chức năng quản lý thư mục

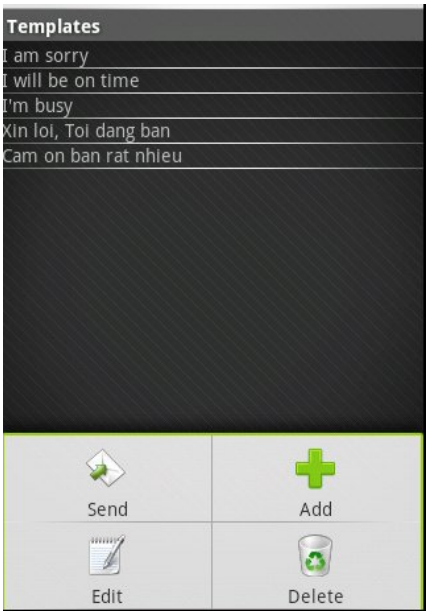
8.2.7. Xóa tin nhắn

Trong màn hình **Delete Screen** người dùng có thể chọn xóa tất cả tin nhắn trong thư mục có trong hệ thống (inbox, send item, save item hay tin nhắn trong các thư mục do người dùng tạo ra.



Hình 8.15: Màn hình xóa tin nhắn

8.2.8. Quản lý template



Hình 8.16 Màn hình quản lý template

Chức năng	Ý nghĩa
Send	Gửi tin nhắn theo mẫu template
Add	Thêm mới template
Edit	Hiệu chỉnh template đang chọn
Delete	Xóa template đang chọn.

Bảng 8.11 Danh sách chức năng template

8.2.9. Quản lý cấu hình hệ thống

Hình 8.17: Màn hình quản lý cấu hình ứng dụng sms

Chức năng	Ý nghĩa
OK	Chấp nhận các thông số như: thuật toán mã hóa, tự động mã hóa khi insert, tự động giải mã khi đăng nhập, thời gian timeout chương trình
Cancel	Hủy các thông số vừa hiệu chỉnh

Bảng 8.12 Danh sách chức năng cấu hình

Các thông tin quản lý cấu hình bao gồm :

- **Algorithm** : thuật toán dùng để mã hóa tin nhắn
- **Encrypt incoming Sms** : có tự động mã hóa khi có tin nhắn đến hay không.
- **Ask key when send sms**: nếu thông số này được chọn, thì khi gởi tin nhắn với phương thức mã hóa, chương trình sẽ xác nhận người dùng có muốn sử dụng khóa đang có trong hệ thống hay muốn nhấp khóa mới để mã hóa tin nhắn

8.3. Kết luận

Trong chương này chúng em đã trình bày tổng quan về kiến trúc ứng dụng GPSms, giới thiệu sơ lược về vai trò của các lớp trong ứng dụng. Đồng thời trình bày tóm tắt các chức năng chương trình hỗ trợ người dùng.

Chương 9

Kết luận

✍ *Nội dung của chương này trình bày các kết quả đạt được và hướng phát triển của đề tài.*

9.1. Môi trường phát triển và thử nghiệm

Đề tài bao gồm hai ứng dụng GPContact (quản lý contact) và GPSms (quản lý tin nhắn). Các công cụ và môi trường được sử dụng để thực hiện đề tài:

- Công cụ phân tích, thiết kế: Rational Rose 2003, Visio 2003
- Môi trường cài đặt ứng dụng: Microsoft Windows XP SP2
- Công cụ lập trình
  - Android SDK phiên bản m5-rc14.
- Môi trường lập trình (IDE):
  - Eclipse 3.3
- Cơ sở dữ liệu:
  - SQLite Database
- Môi trường thử nghiệm và cài đặt:
  - Máy ảo Dalvik for Android.

Cài đặt và thử nghiệm :

- Sử dụng file cài đặt GPContact.apk để cài đặt chương trình GPContact.
- Sử dụng file cài đặt GPSms.apk để cài đặt chương trình GPSms.

Các kết quả thử nghiệm được thực hiện trên máy ảo được cài đặt trên máy có thông tin cấu hình : Win XP SP2, Intel Petium dual-core 1.6GHx, RAM: 512MB)

Ứng dụng GPContact

STT	Tính năng thử nghiệm	Đánh giá
1	Khởi động ứng dụng	Khởi động nhanh thời gian khởi động <1s
2	Load danh sách contact	Thời gian chờ tối đa < 1s (Số lượng rất ít)
3	Thực hiện các thao tác thêm xóa, sửa, cập nhật lại danh sách contact	Nhanh thời gian không đáng kể.
4	Thực hiện cuộc gọi	Chưa thể thực hiện được với máy ảo, chức năng này chỉ giả lập theo nghi thức mạng.

Bảng 9.1 Thử nghiệm ứng dụng GPContact

Ứng dụng GPSms

STT	Tính năng thử nghiệm	Đánh giá
1	Khởi động ứng dụng	Khởi động nhanh, từ khi bắt đầu đến khi hiển thị danh sách từ < 1s
2	Load danh sách tin nhắn SMS	Thời gian chờ tối đa < 5s
3	Load danh sách contact	Chương trình chạy tốt và chưa phát hiện lỗi
4	Chuyển thư mục, mã hóa, giải mã, đọc danh sách	Chương trình thực hiện tốt .
5	Gởi tin nhắn	Chức năng này giả lập theo nghi thức mạng.Cho phép gởi tin nhắn giữa hai máy ảo dựa vào địa chỉ IP, Port của hai máy ảo.

Bảng 9.2 Thử nghiệm ứng dụng GPSms

9.2. Các kết quả đạt được

Dựa trên cơ sở tim hiểu về nền tảng, kỹ thuật lập trình và quy trình bảo mật thông tin trên Android, đề tài đạt được mục tiêu đề ra là xây dựng bộ ứng dụng bảo mật thông tin người dùng trên điện thoại di động chạy hệ điều hành Android.

Bộ ứng gồm 2 chương trình sau:

- Ứng dụng GPContact: quản lý các thông tin Contact, cho phép mã hóa và giải mã thông tin
- Ứng dụng GPSms: quản lý tin nhắn SMS, các chức năng tương tự với ứng dụng SMS trên các điện thoại di động thông thường, cho phép mã hóa để bảo mật thống tin

Ngoài bộ ứng dụng xây dựng được, chúng em đã tim hiểu được về hệ điều hành Android, bao gồm môi trường phát triển, các kỹ thuật xây dựng ứng dụng, xử lý dữ liệu Contact, SMS.

Ngoài ra, chúng em xây dựng hoàn chỉnh úng dụng quản lý tin nhắn (GPSms) có tích hợp nội dung bảo mật thông tin như một đóng góp về ý tưởng **"Ứng dụng quản lý tin nhắn"** cho cộng đồng phát triển ứng dụng trên nền tảng Android (đến thời điểm hiện tại Android chưa có sẵn ứng dụng quản lý tin nhắn).

9.3. Hướng phát triển

Tới thời điểm hiện (tháng 7/2008), bộ ứng dụng chỉ có thể chạy trên máy ảo, chưa được thử nghiệm trên máy thật, nên cần cải tiến ứng dụng để có thể chạy trên thiết bị cụ thể chạy hệ điều hành Android

Phát triển các ứng dụng khác trên nền tảng Android, nhằm quản lý và bảo mật các thông tin như :

- Ứng dụng quản lý Email
- Ứng dụng quản lý File Explorer, hỗ trợ bảo mật tập tin, thư mục
- Ứng dụng ghi chú (Note)
- Hỗ trợ tin nhắn MMS
- Và các ứng dụng khác nhằm bảo mật thông tin người dung.

Phụ lục A Thư viện mã hóa trong Android

A.1 Giới thiệu

Android sử dụng một phần thư viện trong JDK, trong đó có các gói thư viện bảo mật của thư viện bảo mật (JCA – Java Cryptography Architecture, JCE - Java Cryptography Extension). Ngoài ra trong đề tài còn sử dụng gói thư viện mã hóa cryptix-jce .

A.2 Thư viện JCA/JCE

JCA là một môi trường làm việc để có thể truy cập và phát triển các hàm mã hóa trong Java. JCA được thiết kế với hai mục tiêu cơ bản sau:

- Sự phát triển mang tính độc lập và khả năng làm việc cấp cao
- Sự độc lập về thuật toán và khả năng mở rộng.

Với khả năng đọc lập cài đặt và độc lập thuật toán, bạn có thể sử dụng các dịch vụ mã hóa có trong JCA như: Chữ ký điện tử (digital signatures), thông điệp rút gọn (message digests) mà không cần quan tâm đến cài đặt chi tiết của các thuật toán.

Khả năng cài độc lập thuật toán có được bằng cách định nghĩa những loại của dịch vụ mã hóa và định nghĩa các lớp cung cấp hàm cho các dịch vụ mã hóa này. Những lớp này gọi là Engine Class, ví dụ như MessageDigest, Signature, KeyFacroty, KeyPairGenerator và Cipher.

Khả năng độc lập cài đặt được xây dựng dựa trên kến trúc "Provider". Thuật ngữ Cryptigraphic Service Provider (CSP) chỉ ra một gói hay tập hợp các gói cài đặt một hay nhiều dịch vụ mã hóa như chữ ký điện tử, thuật toán rút gọn thông điệp hay dịch vụ phát sinh khóa.

Khả năng làm việc cao cao cấp có nghĩa là nhiều quá trình làm việc có thể tiến hành đồng thời, sử dụng khóa của nhau, hoặc kiểm tra chữ ký của nhau. Điều này cũng rất có giá trị, ví dụ như trong trường hợp sử dụng chung một thuật toán, một khóa được phát sinh bởi một Provider có thể được sử dụng chung và một chứ ký phát sinh bởi một Provider cũng có thể dùng chung cho một cái khác.

JCE (Java Cryptography Extendsion) được cung cấp như mợt phần mở rộng của Java. JCE cung cấp một môi trường làm việc và thực thi mã hóa, phát sinh khóa, chấp nhận khóa và MAC để bỗ 3 sung thêm giao tiếp và triển khai số hóa thông điệp, chữ ký số.

Kiến trúc Provider của JCA cho phép độc lập giải thuật. JCE cũng chia sẽ cùng một lý thuyết của triền khai và độc lập thuật toán bằng cách sử dụng Provider. Nhằm mục đích sử dụng các thuật toán mới trong việc phát sinh khóa, JCE cũng đưa ra một số interface và lớp để có thể triển khai các khái niệm này.

JCE cung cấp các phương thức cho mã hóa đối xứng thông qua việc sử dụng khóa bí mật, một khóa được chia sẽ bởi người gởi và ngừoi nhận để mã hóa cũng như giải mã dữ liệu.

A.3 Kiến trúc Provider (*Cryptographic Service Providers)*

Java.security.Provider là một lớp cơ sở cho tất cả các nhà cung cấp bảo mật. Mỗi CSP **(**Cryptographic Service Providers) là một thể hiện của lớp này bao gồm tên nhà cung cấp và liệt kê tất cả những dịch vụ/thuật toán mà nó (CSP) cài đặt. Khi cần một thể hiện của thuật toán cụ thể, kiến trúc JCA sẽ thăm dò cơ sở dữ liệu "Provider", và nếu tìm được một CSP thích hợp thì thể hiện CSP này sẽ được khởi tạo

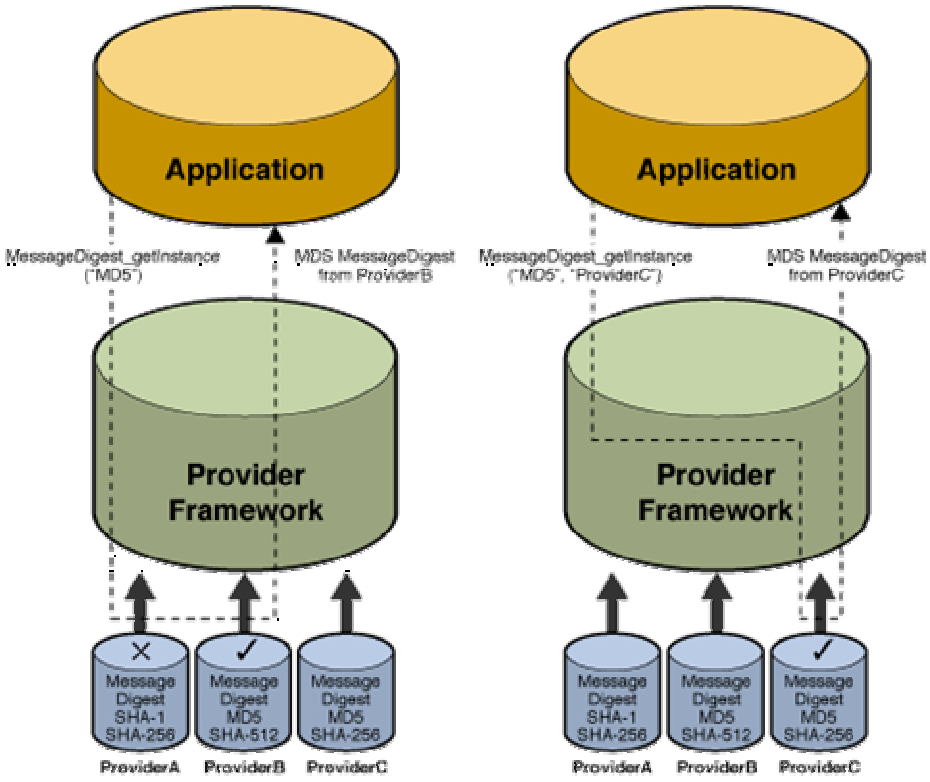
Các CSP chứa một gói (hoặc một tập các gói) cung cấp các cài đặt cụ thể cho các thuật toán bảo mật mà CSP này quảng cáo. Mỗi phiên bản JDK sẽ có một hay nhiều CSP kèm theo. Các CSP khác có thể được thêm tĩnh hoặc động. Người dùng hoàn toàn có thể cấu hình môi trường nhằm chỉ rõ thứ tự ưu tiên cho các CSP. Thứ tự này sẽ là cơ sở tìm kiếm khi có yêu cầu mà không chỉ rõ cụ thể CSP nào.

Khi dùng kiến trúc JCA, một ứng dụng đơn giản chỉ yêu cầu một loại đối tượng cụ thể (như một thông điệp rút gọn) và một thuật toán hay một dịch vụ cụ thể (như thuật toán "MD5") và nhận một cài đặt từ một trong các CSP. Hoặc ứng dụng có thể yêu cầu trực tiếp một CSP. Mỗi CSP được đại diện bằng tên

md = MessageDigest.getInstance("MD5");

md = MessageDigest.getInstance("MD5","ProviderC");

Hình sau minh họa việc yêu cầu một cài đặt của thông điệp rút gọn "MD5". Hình cho thấy có 3 CSP khác nhau cài đặt các thuật toán rút gọn thông điệp ("SHA-1", "MD5", "SHA-256"). Các CSP được sắp thứ tự ưu tiên từ trái qua phải. Trong minh họa đầu tiên, một ứng dụng yêu cầu cài đặt thuật toán MD5 nhưng không cho biết tên CSP cụ thể. Các CSP sẽ được tìm kiếm theo thứ tự ưu tiên và cài đặt của CSP đầu sẽ được ứng dụng sử dụng, CSP B sẽ được trả về. Trong hình 2 thì ứng dụng sẽ yêu cầu cài đặt của thuật toán MD5 từ một CSP cụ thể là CSP C. Vào lúc này thì cài đặt thuật toán của CSP C sẽ được dùng, cho dù theo thứ tự ưu tiên thì CSP B cao hơn C.



Hình A.1 Kiến trúc thu viện JCA

JCA đưa ra một tập các APIs cho phép người dùng truy vấn CSP nào đã được cài đặt và các dịch vụ gì được hỗ trợ. Kiến trúc này cũng giúp người dùng cuối dễ

dàng thêm vào các CSP phụ. Nhiều bên thứ ba cũng đã cung cấp các cài đặt thuật toán.

A.4 Thư viện Cryptix

Android sử dụng lại các gói thư viện JCA/JCE, tuy nhiên phần cài đặt Provider để hỗ trợ các thuật toán mã hóa đối xứng và bất đối xứng chỉ mới hỗ trợ các thuật toán như DES,AES, DESede.

Cryptix là thư viện cài đặt nhiều thuật toán mã hóa đối xứng .Cryptix là thư viện mã nguồn mở (http://www.cryptix.org/). Cryptix hỗ trợ các thuật toán mã hóa đối xứng sau:

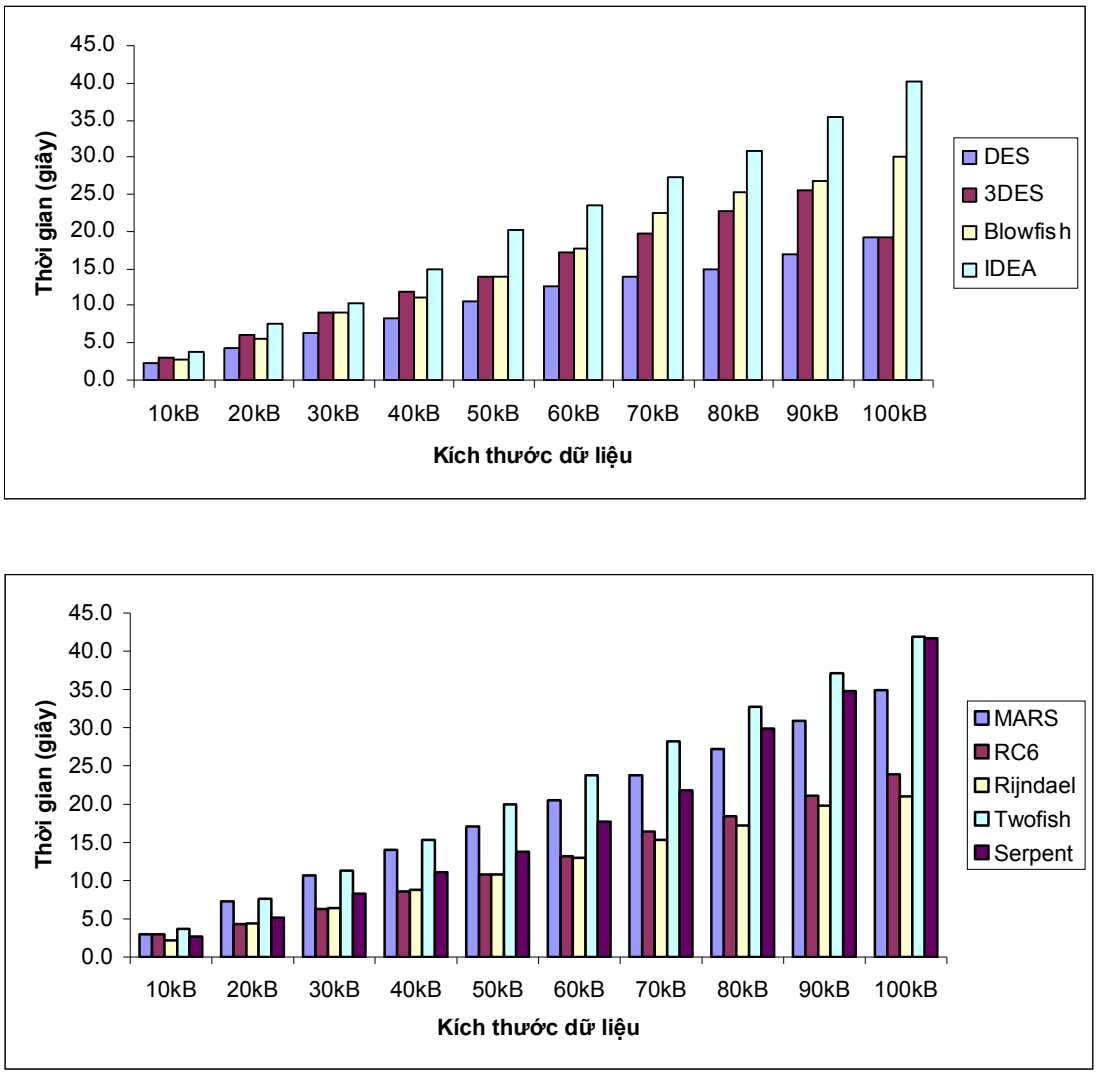
Tên thuật toán	Kích thước khóa	Chiều dài khối
Blowfish	32 - 448	64
DES	56	64
IDEA	128	64
MARS	128/192/256	128
RC6	128/192/256	128
Rijndael	128/192/256	128
Serpent	128/192/256	128
TripleDES	168	64
AES	128/192/256	128

Bảng A.1: Danh sách các thuật toán hỗ trợ bởi Cryptix

Phụ lục B Thử nghiệm các thuật toán mã hóa

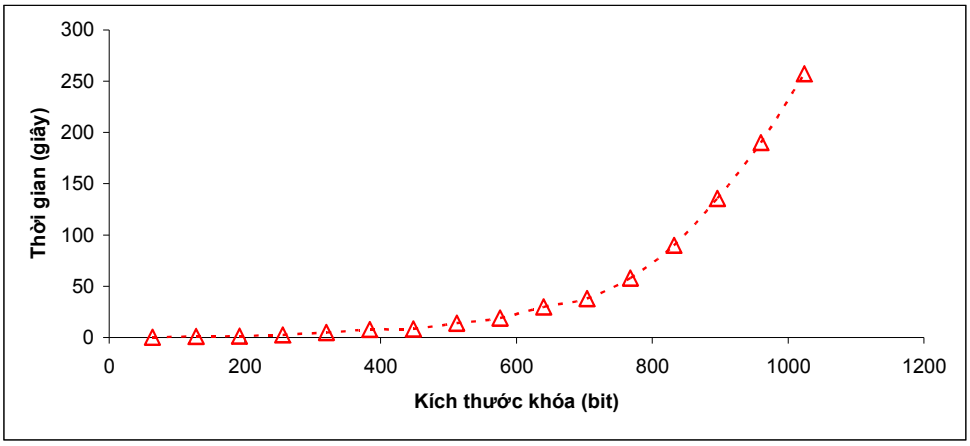
Các kết quả thử nghiệm được thực hiện trên máy ảo được cài đặt trên máy có thông tin cấu hình : Win XP SP2, Intel Petium dual-core 1.6GHx, RAM: 512MB

- Thử nghiệm thời gian mã hóa của một số thuật toán mã hóa đối xứng:



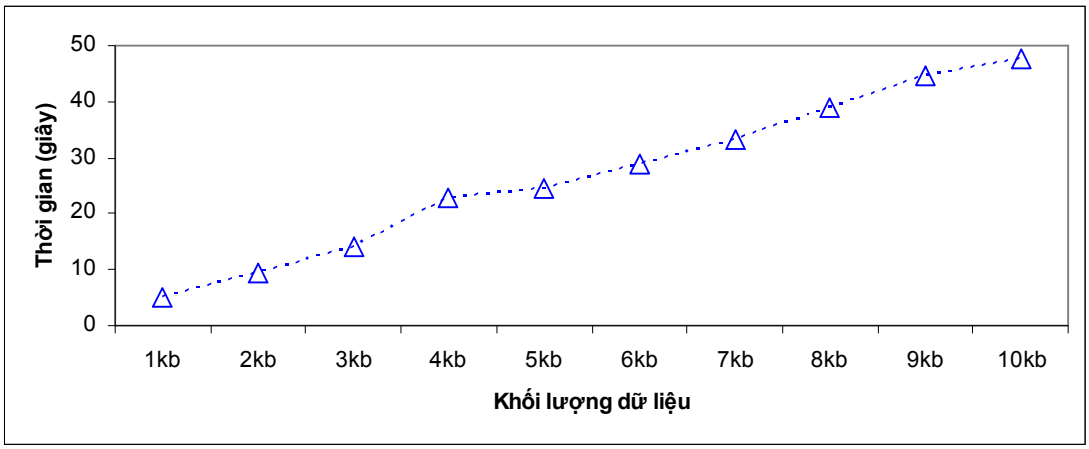
Hình B.1: Thử nghiệm mã hóa với các thuật toán đối xứng.

- Thử nghiệm thời gian tạo khóa với thuật toán RSA



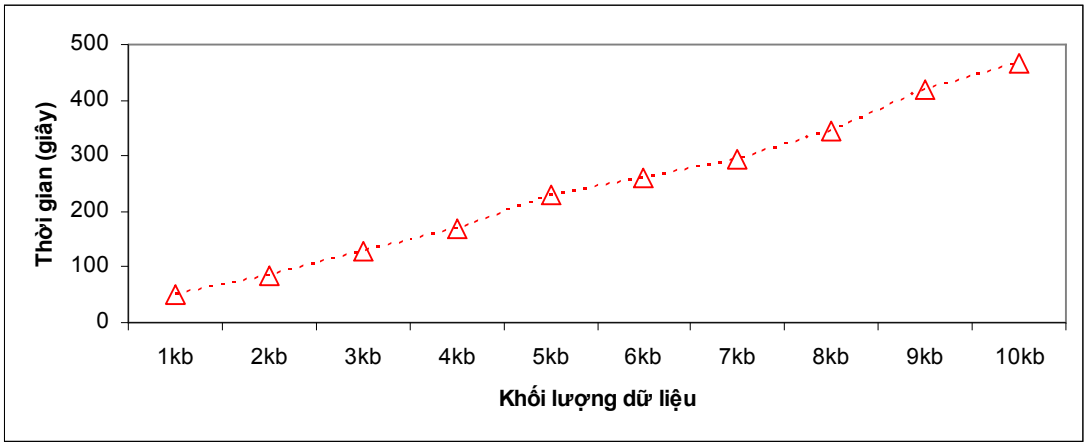
Hình B.2: Thử nghiệm thời gian tạo khóa với thuật toán RSA

- Thử nghiệm mã hóa và giải mã thuật toán RSA, kích thước khóa 1024



Hình B.3: thử nghiệm mã hóa với thuật toán RSA (kích thước khóa 1024)

- Thử nghiệm giải mã với thuật toán RSA, kích thước khóa 1024



Hình B.4: Thử nghiệm giải mã với thuật toán RSA (kích thước khóa 1024).

Tài liệu tham khảo

[1] PGS. TS. Dương Anh Đức, Th.S. Trần Minh Triết, *Mã hóa và ứng dụng*, NXB Đại học Quốc gia TP. HCM, 2005.

[2] Hồ Kim Huy, Hoàng Minh Châu, *Nguyên cứu và xây dụng ứng dụng bảo vệ thông tin trên môi trường Symbian*, Luận văn cử nhân tin học, Đại học Khoa học Tự nhiên Tp.Hồ Chí Minh, 2007.

[3] David Hook, *Beginning Cryptography with Java*, Wrox Press © 2005

[4] Jonathan Knudsen, *Java Cryptography***,** O'REILLY

[5] Tài liệu hướng dẫn đính kèm trong SDK Android phiên bản m5-rc14