INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS" ZACATENCO

"APLICACIÓN ANDROID PARA ENCRIPTACIÓN DE DOCUMENTOS MEDIANTE EL RECONOCIMIENTO DE VOZ"

TESIS

PARA OBTENER EL TÍTULO EN

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTA:

KASSANDRA MONTSERRAT GARDUÑO PIÑA

ASESORES

DR. MARIO JIMÉNEZ HERNÁNDEZ

DR. JUAN PABLO FRANCISCO POSADAS DURÁN

Ciudad de México a 16 de Abril de 2019

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TÍTULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	**TESIS Y EXAMEN ORAL INDIVIDUAL**
DEBERA (N) DESARROLLAR	**C. KASSANDRA MONTSERRAT GARDUÑO PIÑA**

"APLICACIÓN ANDROID PARA ENCRIPTACIÓN DE DOCUMENTOS MEDIANTE EL RECONOCIMIENTO DE VOZ"

DISEÑAR E IMPLEMENTAR UN PROTOTIPO DE APLICACIÓN ANDROID QUE PERMITA LA ENCRIPTACIÓN Y DESENCRIPTACIÓN DE DOCUMENTOS USANDO UN ALGORITMO DE ENCRIPTACIÓN RC4 Y QUE PERMITA EL ACCESO A LA APLICACIÓN MEDIANTE EL RECONOCIMIENTO DE VOZ.

- **INTRODUCCIÓN**
- **ESTADO DEL ARTE**
- **MARCO TEÓRICO**
- **DISEÑO E IMPLEMENTACIÓN**
- **PRUEBAS Y RESULTADOS**
- **CONCLUSIONES Y TRABAJO A FUTURO**

CIUDAD DE MÉXICO, A 16 DE ABRIL DE 2019.

A S E S O R E S

DR. JUAN PABLO FRANCISCO POSADAS DURÁN

DR. MARIO JIMÉNEZ HERNÁNDEZ

ING. GABRIEL VEGA REYES
JEFE DE LA CARRERA DE INGENIERÍA
EN COMUNICACIONES Y ELECTRÓNICA

Autorización de uso de obra

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad la que suscribe **KASSANDRA MONTSERRAT GARDUÑO PIÑA**, manifiesto ser autora y titular de los derechos morales y patrimoniales de la obra titulada **"APLICACIÓN ANDROID PARA ENCRIPTACIÓN DE DOCUMENTOS MEDIANTE EL RECONOCIMIENTO DE VOZ",** en adelante **"La Tesis"** y de la cual se adjunta copia un impreso y un cd por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo al **Instituto Politécnico Nacional,** en adelante **El IPN**, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales o en cualquier otro medio; para apoyar futuros trabajos relacionados con el tema de **"La Tesis"** por un periodo de **5 años** contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a **El IPN** de su terminación.

En virtud de lo anterior, **El IPN** deberá reconocer en todo momento mi calidad de autora de **"La Tesis"**.

Adicionalmente, y en mi calidad de autora y titular de los derechos morales y patrimoniales de **"La Tesis"**, manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto de **"La Tesis"**, por lo que deslindo de toda responsabilidad a **El IPN** en caso de que el contenido de **"La Tesis"** o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México., a 16 de abril de 2019.

Atentamente

Kassandra Montserrat Garduño Piña

Índice General

Índice de Figuras

Índice de Tablas

CAPÍTULO 1. Introducción

En la actualidad existen casi 7.5 mil millones de habitantes en el mundo y en 2016 se contabilizaron 2.6 mil millones de personas con un teléfono inteligente y aunque eso es menos de la mitad de la población mundial, las cifras crecen rápidamente. De acuerdo con proyecciones de Ericsson, en 2019 habrán 6.1 mil millones de personas conectadas a internet desde un teléfono inteligente, es decir, el 70% de la población del planeta.

El procesamiento y la administración de datos personales es imprescindible para la realización de diversas actividades cotidianas de personas, empresas, asociaciones y organizaciones, por ejemplo, en los últimos años el uso de la banca electrónica en México ha registrado un importante crecimiento entre algunos sectores de la población. De acuerdo con datos de la Fundación de Estudios Financieros del Instituto Tecnológico Autónomo de México (ITAM), actualmente existen 12 millones de internautas[1] bancarizados y de éstos, aproximadamente el 75% utiliza la banca electrónica para revisar sus saldos, en tanto que un 50% hace operaciones de transferencias. Del total de transacciones bancarias que se hacen al día, apenas un 16% se realiza a través de internet y sólo alrededor del 7% de los internautas utiliza un teléfono inteligente para llevar a cabo una operación bancaria[2].

La mayoría de los datos se guardan en formato digital, los archivos informáticos pueden contener información de todo tipo: texto, imágenes, sonido o video. Algunos de los riesgos de guardar la información en forma digital son: la posible pérdida de información por avería del equipo informático; robo del equipo informático puede revelar información confidencial; ataques mediante virus.

La encriptación o cifrado de archivos, es un procedimiento que vuelve completamente ilegibles los datos de cualquier archivo, provocando que el archivo se vuela prácticamente inservible para un usuario no autorizado, ya que, sin la contraseña, incluso si lo ha interceptado o lo ha copiado, no podrá leerlo o visualizarlo.

El desarrollo de una aplicación Android para teléfonos móviles que permita encriptar los archivos personales de los usuarios puede ayudar a evitar el mal uso de la información personal obtenida de estos equipos al ser robados o perdidos. Una forma de aumentar la seguridad es utilizara el reconocimiento de voz como clave en lugar de contraseñas escritas que pueden ser descifradas con equipos especiales. El reconocimiento de voz es una manera más confiable de autenticar a una persona porque se basa en una característica de la persona y no en una secuencia aleatoria de caracteres, proporcionando una experiencia sin fisuras y mucho más rápida que las opciones manuales, como la introducción de la contraseña. Adicionalmente, ofrece la ventaja para los usuarios de evitar el tener que recordar varias contraseñas, números de identificación y respuestas de seguridad.

[1] http://dle.rae.es/?id=LvqsKVk

[2] http://www.condusef.gob.mx/Revista/index.php/usuario-inteligente/servicios-financieros/675-banca-electronica-vs-banca-tradicional

La voz de una persona se compone de más de 100 características que se definen por la configuración física de la boca y la garganta. Esto significa que la voz de todos es diferente, y que su voz es tan única como su huella digital; de hecho aún más, permitiéndonos mayor seguridad.

1.1 Planteamiento del Problema

El incremento en las capacidades y prestaciones de los celulares inteligentes ocasiona que sea cada vez más frecuente que los usuarios manejen información confidencial y la almacenen en ellos, como sus datos personales. Los datos personales se refieren a toda aquella información relativa al individuo que lo identifica, le dan identidad. Algunos de ellos son nombre, domicilio, teléfono, correo electrónico, firma, RFC, CURP, fecha de nacimiento, edad, nacionalidad, entre otros.

El robo de estos celulares en la Ciudad de México se duplicó en el primer semestre de 2017 con respecto al mismo periodo del año pasado, de acuerdo con cifras de la Procuraduría General de Justicia. Entre enero y mayo del 2017 se han registrado 4 mil 399 denuncias por robo de celular con y sin violencia, tan sólo en el mes de marzo se iniciaron 915 averiguaciones previas por este delito[3].

Con la información obtenida de un celular se pueden realizar numerosas actividades fraudulentas, como lo son el contratar créditos o servicios a tu nombre, realizar operaciones inmobiliarias, acceder a cuentas bancarias, publicar tus datos y enviar información a tu lista de contactos.

Debido a lo expuesto anteriormente, se propone crear una aplicación Android que permita la encriptación y desencriptación de archivos confidenciales usando el reconocimiento de voz.

1.2 Objetivo General

Diseñar e implementar un prototipo de aplicación Android que permita la encriptación y desencriptación de documentos usando un algoritmo de encriptación RC4 y que permita el acceso a la aplicación mediante el reconocimiento de voz.

1.3 Objetivos Específicos

- ✓ Implementar un módulo para la encriptación y desencriptación de documentos usando un algoritmo de encriptación.
- ✓ Implementar un módulo para el reconocimiento de voz.
- ✓ Diseñar una interfaz para la aplicación
- ✓ Implementar una interfaz para el manejo de documentos
- ✓ Realizar pruebas de la aplicación

[3] https://www.reporteindigo.com/reporte/robo-de-celulares-a-la-alza-en-2017/

1.4 Justificación

Según Sundar Pichai, CEO de Google, Android ya ha superado la barrera de los mil millones de usuarios, lo que supone que ocho de cada diez personas con Smartphone han apostado por su sistema operativo.

El sistema operativo Android de Google se convirtió en el más utilizado en los teléfonos del mundo en el tercer trimestre de este año, ampliando su ventaja sobre el iOS de Apple y el Windows Phone de Microsoft. Los números publicados por StatCounter muestran que Android tiene el 37.93% del uso mundial de los sistemas operativos que se conectan a Internet, a partir de marzo del 2017.

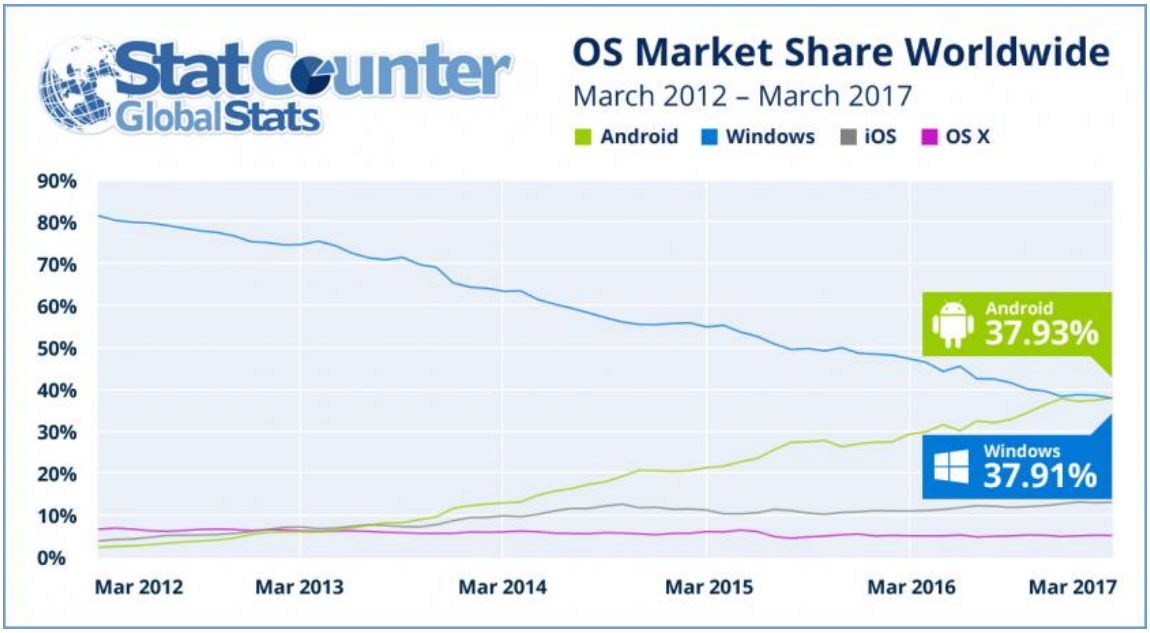

Figura 1.1 Porcentaje de usuarios con dispositivos Android.

En México, la cantidad de ciudadanos que usa teléfono celular creció 20 por ciento en el último año, al pasar de 50.6 millones a los recientes 60.6 millones. El 82.5% de los teléfonos celulares en nuestro país cuentan con Android, dejando muy atrás al siguiente contendiente, iOS, con solo el 10.3% de la cuota de mercado.

Considerando que, a nivel nacional, 65.5 millones de personas hacen uso de internet, el INEGI[4] reporta que las principales actividades hechas en un teléfono inteligente son el uso de redes sociales, mensajes de texto, obtención de información, operaciones bancarias en línea e interacción con portales gubernamentales

[4] http://www.inegi.org.mx/saladeprensa/aproposito/2017/internet2017_Nal.pdf

Teniendo en cuenta que aproximadamente 90 de cada 100 personas están suscritas a una compañía de telefonía móvil, las estadísticas de robos siguen creciendo. Es cierto que cada vez más usuarios tienen más aplicaciones de seguridad como rastrear un celular, bloquear el teléfono o reportar un robo, pero esto en muchas ocasiones no es suficiente para proteger los datos personales de los usuarios.

Una solución a esto es la encriptación, proceso que vuelve ilegible información considerada importante, que una vez encriptada sólo puede leerse aplicándole una clave. Se trata de una medida de seguridad que es usada para proteger información delicada que no debería ser accesible a terceros.

En el presente trabajo se justifica el uso de Android dado el hecho de que al estar liberado con licencia Apache y código abierto lo convierte en un sistema operativo totalmente libre para que un desarrollador no solo pueda modificar su código sino también mejorarlo.

Dado que puede instalarse prácticamente en todo tipo de dispositivos, siempre está presente en los terminales más potentes del mercado, mientras que otro tipo de sistemas operativos se ven obligados a estar rezagados a terminales más obsoletos o estar limitados a una determinada marca de fabricante.

Además podemos cubrir una mayor cantidad de usuarios, dando énfasis a que Android cuenta con 2.000 millones de dispositivos activos al mes alrededor del mundo, cifra que hasta hoy sigue en aumento.

CAPITULO 2. Estado del Arte

2.1. Métodos para Encriptar Dispositivos Informáticos

Si se encripta el contenido de un dispositivo nadie podrá acceder a la información que contiene. La encriptación es una estrategia de seguridad más robusta que una contraseña en la pantalla de inicio del móvil o el PC, porque esta última que sólo protege el acceso, pero no los datos.

A continuación se describen algunos ejemplos en los que se utilizan técnicas de encriptación.

- **Ordenadores:** Con Windows Vista, Microsoft estrenó un sistema de encriptación propio llamado BitLocker. Utiliza el algoritmo AES con claves de 128 o 256 bits.En la práctica sólo los ordenadores más nuevos y con cierta combinación de hardware permiten a Windows encriptar o cifrar los discos duros automáticamente.
- **Android:** Google ha prometido que Android 6.0 incluirá la encriptación por defecto. Android 5.0 Lollipop permite utilizarla, pero en la mayoría de los casos debe llevarla a cabo el usuario. Sólo algunos smartphones Android nuevos, como los últimos Nexus, ya vienen encriptados de fábrica.
- **iOS:** Desde iOS 8 Apple encripta automáticamente la memoria interna de los dispositivos iPhone y iPad, así que todos los datos ya están protegidos. Eso le ha acarreado numerosas presiones y reproches del gobierno norteamericano, pero la Compañía de la Manzana no ha cedido.
- **iOS 9**: Utiliza un criptomotor con el algoritmo de encriptación AES de 256 bits, uno de los más avanzados, situado entre la memoria Flash y la memoria virtual. Desde el iPhone 3GS cada dispositivo de Apple posee un identificador único UID almacenado en un chip, que no conoce nadie, ni siquiera Apple. Es el que se utiliza para encriptar y desencriptar los datos. Además, el propio sistema operativo añade varias capas de protección por clase asociada al código de desbloqueo del dispositivo.

2.2. Tecnología Biométrica de Voz

Todos los días se pueden leer noticias sobre algún ataque exitoso a sistemas informáticos. A menudo, sistemas de control crítico están en la red, protegidos únicamente por una contraseña débil, o incluso con la contraseña por defecto de fábrica. El comportamiento erróneo del componente humano y la ausencia de validaciones se traduce en aplicaciones vulnerables a ataques externos.

Descifrar contraseñas de ocho caracteres mediante ataques de fuerza bruta es posible en unas 5,5 horas. Con una máquina dedicada a tal fin, y empleando unidades de procesamiento gráfico; el precio de estas máquinas ronda los $30,000.

La longitud de la contraseña no es la única preocupación. El factor humano expone el proceso de gestión de contraseñas a un riesgo muy serio. La mayoría de las personas no tienden a recordar contraseñas largas y complejas, al revés, usan contraseñas fáciles de recordar y afines a su vida cotidiana.

En muchos casos, las contraseñas se reutilizan en varios servicios, desde cuentas de Facebook a la bancaria. La media de usuarios tiene unas 26 cuentas protegidas con contraseña, pero solamente 5 son distintas. De acuerdo con un estudio reciente basado en seis millones de contraseñas generadas por el usuario, las 10.000 contraseñas más comunes eran usadas en el 98.1% de las cuentas. Información que nos da una idea de cómo es de vulnerable la gestión de credenciales.

Además, otro problema referente a las contraseñas viene dado por la plataforma que se use. Si se considera que la media de usuarios tarda unos 4-5 segundos en escribir una contraseña "fuerte" de diez caracteres en un teclado convencional. Unos 7-10 segundos en un dispositivo móvil con teclado hardware y de 7 a 30 segundos en pantallas táctiles, un cuarto de la gente encuestada admitió usar contraseñas menos seguras para ahorrar tiempo.

Es por ello que algunas empresas están promoviendo la voz humana como medida para asegurar las cuentas de los usuarios y que remplazar a los sistemas tradicionales de autenticación de contraseñas que pueden ser fáciles de hackear.

Ya existen bancos e instituciones financieras que han implementado esta tecnología biométrica de voz para verificar las identidades de sus clientes. La tecnología se desplegó por primera vez en un centro de atención al cliente en 2001. A partir de ahí, también ha sido usada para aplicaciones móviles relacionadas con finanzas y con la seguridad de los PC.

Factores como la laringe de una persona, la forma de la cavidad nasal o la falta de dentadura, determinará la forma en la que suena una persona. La gente también puede hablar de una manera más monótona o más viva, o espaciar sus palabras en ritmos variables. Las huellas digitales permiten hasta 20 puntos de medición, mientras que nuestra voz y los algoritmos que se pueden utilizar permiten 1000 puntos. La forma en que pronunciamos las palabras o nuestro tono de voz, también permiten que la identificación sea más efectiva. La ventaja de utilizar nuestra voz (que es única) como contraseña, reduce las posibilidades de que hackers accedan a nuestros datos.

2.3. Seguridad en Android

La consolidación de Android como el sistema operativo más utilizado en los dispositivos móviles de la actualidad lo ha convertido en blanco de diversas amenazas que continúan creciendo, una tendencia que se ha visto en los últimos años.

De acuerdo con IDC[5], hasta el tercer trimestre de 2014, Android ocupaba el 84.4% del mercado de dispositivos móviles. Actualmente se estima que el número de *smartphones* y tabletas que tienen instalado este sistema operativo, supera los mil millones y que su uso seguirá en aumento, ya que ahora será posible encontrarlo en otros dispositivos inteligentes como relojes, televisores e incluso en automóviles.

[5] International Data Corporation

De forma paralela, la cantidad y diversidad de la información que los usuarios almacenan, procesan o transmiten a través de estos dispositivos, cuando utilizan servicios de Internet como correo electrónico, redes sociales o banca en línea, contribuye a aumentar su importancia y en consecuencia, el interés de los cibercriminales por la plataforma móvil. Como resultado, en los últimos años se ha observado un incremento en el desarrollo de software malicioso que tiene como objetivo este popular sistema operativo.

La seguridad en Android es un tema que preocupa tanto a Google, el dueño del sistema operativo, como a todos sus usuarios, aunque cada uno por motivos distintos. El usuario quiere tener la certeza de que sus datos están a salvo, mientras que los de Mountain View quieren que el sistema operativo sea seguro para que las empresas confíen en sus servicios.

Los teléfonos con sistema operativo Android, a medida que avanza la tecnología, son capaces de almacenar más y más datos, ya sean fotos, vídeos, archivos de texto, información de nuestra economía, y así casi hasta el infinito. Es por eso que una de las características que introdujo Google hace unos años es la de poder encriptar nuestro teléfono.

Con encriptar o cifrar el contenido de un móvil se refiere a hacer que la información que hay dentro del dispositivo sea mucho menos accesible de lo habitual. Esto se consigue haciendo que para llegar a cierta información que se tenga almacenada, se debe introducir una contraseña, de forma que si el teléfono cae en malas manos, tengas cierta tranquilidad de que tus datos no están tan expuestos.

Una vez cifrado el móvil, la música, vídeos, fotos y datos de aplicaciones sólo serán accesibles si introduces la contraseña o el código PIN que has configurado antes de iniciar el proceso.

Esto no quiere decir que sea imposible que personas malintencionadas puedan acceder a esos datos, simplemente que será mucho más difícil, tanto que sólo podrá conseguir lo que quiere si sabe tu contraseña. Ni siquiera si consigue extraer los datos al pasarlos al PC o de alguna otra manera podrá entender lo que tiene, sólo podrá hacerlo si consigue la clave correcta.

2.3.1. Aplicaciones Android para Encriptación de Archivos

La seguridad de dispositivos Android es una de las cosas que más preocupa a los usuarios de dispositivos móviles sobre todo con todas las noticias que surgen relativas a los errores de seguridad de las aplicaciones o a la cantidad de virus que pueden tener. También es posible que los celulares contengan fotos o archivos que los usuarios no quieren que otras personas vean, pero al mismo tiempo no se puede evitar tener que prestar el smartphone. Por todos estos motivos, lo mejor que se puede hacer es encriptar archivos y carpetas con datos sensibles. Es por ello que dentro de la Play Store existen aplicaciones que realizan este trabajo.

1) SSE Universal Encryption App

Figura 2.1 Logotipo de SSE Universal Encryption App.

Su principal función es proteger con una contraseña cualquier archivo o carpeta de cualquier tipo, como fotos, archivos de audio, de texto, etc.

➤ Password Vault: almacena y administra todas las contraseñas, PIN y notas en un lugar seguro protegido por una contraseña maestra. La función Importar / Exportar está disponible con el formato de archivo comprimido, completamente encriptado .pwv o formato de archivo .xml editable y sin encriptar.

➤ Encriptar Texto: Mantenga sus mensajes, notas y otros textos a salvo de todos los lectores no deseados. Use la base de datos interna o simplemente copie / pegue desde / hacia sus aplicaciones favoritas. Se establece una contraseña para la sesión de cifrado / descifrado actual, y puede tener un número ilimitado de contraseñas para cualquier propósito (notas, correos electrónicos, redes sociales, comunicación con personas, etc).

➤ Encriptar Archivos: Encripte de manera segura sus archivos privados y confidenciales o carpetas enteras.

➤ Algoritmos: Todo está encriptado usando algoritmos de cifrado fuertes: AES (Rijndael) 256 bits, RC6 256 bits, Serpiente 256 bits, Blowfish 448 bits, Twofish 256 bits, GOST 256 bits + Threefish 1024 bits y SHACAL-2 512 bits (para S.S.E versión Pro) están disponibles.

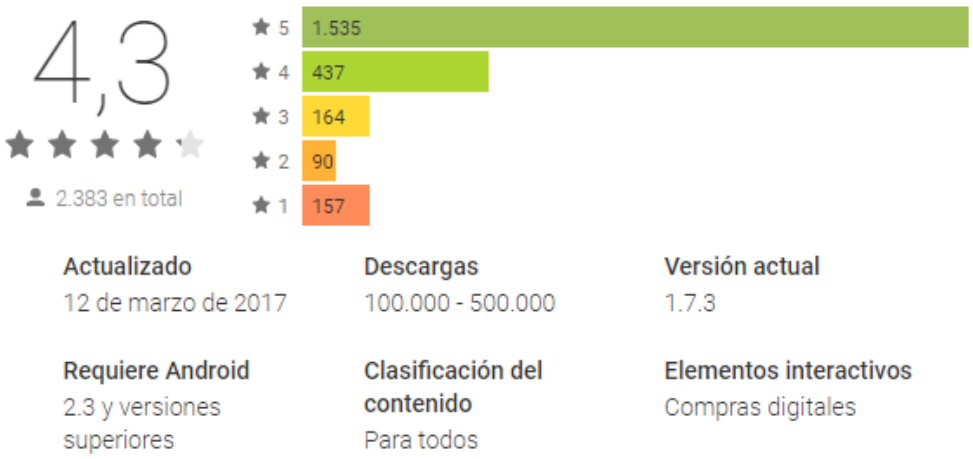

Figura 2.2 Calificación de SSE Universal Encryption App por los usuarios en PlayStore.

Tabla 2.1 Ventajas y desventajas de la aplicación *SSE Universal Encryption App*

Ventajas	**Desventajas**
Encriptación por archivos en lugar de carpetas.	Solo se encuentra en inglés.
Ocupa poco espacio.	No permite selección múltiple de archivos.
Multiplataforma.	
Diferentes sistemas de cifrado.	

2) Encryption Manager

Figura 2.3 Logotipo de Encryption Manager

Da la facilidad de encriptar archivos para su modificación y que la propia app se encarga de volver a encriptar una vez guardado, borrando cualquier rastro. Se puede acceder a los archivos encriptados desde dentro de la propia app o en el navegador de archivos del smartphone, introduciendo la contraseña de cada uno individualmente.

Una contraseña maestra se usa tanto para acceder a la aplicación y para cifrar las claves de cifrado, que se generan al azar para cada archivo, que es administrado por la aplicación. Los archivos están accesibles directamente después del inicio de sesión. Con un clic el archivo se descifra a su ubicación original y puede ser mostrar por el espectador o redactor. Cuando haya terminado de trabajar con la copia descifrada, el archivo se vuelve a cifrar con un solo clic y el archivo descifrado se limpia desde la tarjeta SD. Este proceso limpia los datos con bytes aleatorios antes de eliminar el archivo. Así que incluso si el dispositivo se pierde o es robado, no es posible acceder a los datos confidenciales.

Características:

*Cifratodo tipo de archivos.

* Las funciones básicas de un administrador de archivos (vista en clic, enviar/compartir), pero con el descifrado automático antes de la acción.

* Ofrece cifrado AES y Twofish con 128 y 256 teclas bits.

* Muestra iconos para indicar si un archivo está actualmente descifrar o fue cambiado.

* Ajuste del usuario para la re-encriptación automática al salir.

* Sobrescritura segura del archivo original después de la encriptación.

Figura 2.4 Calificación de "Encryption Manager" por los usuarios en PlayStore.

Tabla 2.2 Ventajas y desventajas de la aplicación *Encryption Manager*.

Ventajas	**Desventajas**
Almacenamiento de contraseñas en archivos encriptados.	No es gratuito.
Solo encripta archivos dentro de la memoria interna.	Número limitado de archivos que pueden encriptarse.
Perdida de datos al desinstalar la aplicación.	Se encuentra en inglés.

3) Encrypt It

Una utilidad multipropósito para encriptar cualquier texto. Utiliza el cifrado de mejores prácticas para garantizar que sus datos estén a salvo de miradas indiscretas. Encripta tus correos electrónicos, mensajes, memos, notas, citas, etc.

- Utiliza el cifrado AES de 256 bits
- Utiliza la salazón para mejorar la seguridad de la semilla / contraseña para resistir ataques de fuerza bruta y tablas arcoiris.
- Utiliza salazón para aleatorizar la salida para mejorar aún más la seguridad.
- Procesamiento rápido y eficiente.
- Texto codificado que puede guardarse en una aplicación o enviarse a alguien.

Para encriptar:
1. Escriba o pegue el texto que se va a cifrar.
2. Ingrese una semilla (contraseña), y presione el botón Encriptar.

Para descifrar:
1. Pegue el texto codificado en la aplicación.
2. Ingrese la semilla (contraseña) y presione el botón Descifrar.

Figura 2.5 Calificación de Encrypt It por los usuarios en PlayStore.

Tabla 2.3 Ventajas y desventajas de la aplicación *Encrypt It*.

Ventajas	Desventajas
Es gratuita.	Solo encripta texto.
Facilidad de uso.	Se encuentra en inglés.

CAPÍTULO 3. Marco Teórico

3.1. Dispositivos Android

3.1.1. Características del Sistema Operativo Android

Android es un sistema operativo creado por la prestigiosa compañía Google, el cual está pensado y desarrollado desde la ideología *OpenSource*, de ahí su enorme éxito y gran aceptación en el relativo poco tiempo de vida que tiene.

Está basado en GNU Linux y enfocado a dispositivos móviles de todo tipo, ya sean teléfonos móviles, tabletas e incluso mini ordenadores portátiles, que ya podemos encontrar en el mercado. Se encarga de recibir las órdenes de los usuarios, de gestionarlas y transmitirlas a los diferentes componentes (hardware) del terminal en donde esté instalado.

Tabla 3.1 Características del sistema operativo Android.

Características	**Descripción**
Plataforma realmente abierta	Es una plataforma de desarrollo libre basada en Linux y de código abierto. Una de sus grandes ventajas es que se puede usar y customizar el sistema sin pagar *royalties*.
Adaptable a cualquier tipo de hardware	Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Hoy en día podemos encontrar relojes, cámaras, electrodomésticos y gran variedad de sistemas empotrados que se basan en este sistema operativo.
Portabilidad asegurada	Las aplicaciones finales son desarrolladas en Java lo que nos asegura que podrán ser ejecutadas en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.
Arquitectura basada en componentes inspirados en Internet	Por ejemplo, el diseño de la interfaz de usuario se hace en XML, lo que permite que una misma aplicación se ejecute en un reloj de pantalla reducida o en un televisor.
Gran cantidad de servicios incorporados	Por ejemplo, localización basada tanto en GPS como en redes, bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia, etc.
Aceptable nivel de seguridad	Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su rango de actuación. Desde La versión 6.0 el usuario puede conceder o retirar permisos a las aplicaciones en cualquier momento.
Optimizado para baja potencia y poca memoria	En el diseño de Android se ha tenido en cuenta el *hardware* específico de los dispositivos móviles. Por ejemplo, Android utiliza la Máquina Virtual ART. Se trata de una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles.

3.1.2. Historia

Google adquiere Android Inc. en el año 2005. Se trataba de una pequeña compañía, recién creada, orientada a la producción de aplicaciones para terminales móviles. Ese mismo año empiezan a trabajar en la creación de una máquina virtual Java optimizada para móviles (Dalvik VM).

En el año 2007 se crea el consorcio Handset Alliance con el objetivo de desarrollar estándares abiertos para móviles. Está formado por Google, Intel, Texas Instruments, Motorola, T-Mobile, Samsung, Ericson, Toshiba, Vodafone, NTT DoCoMo, Sprint Nextel y otros. Una pieza clave de los objetivos de esta alianza es promover el diseño y difusión de la plataforma Android. Sus miembros se han comprometido a publicar una parte importante de su propiedad intelectual como código abierto bajo licencia Apache v2.0.

En noviembre de 2007 se lanza una primera versión del Android SDK. Al año siguiente aparece el primer móvil con Android (T-Mobile G1). En octubre, Google libera el código fuente de Android, principalmente bajo licencia de código abierto Apache (licencia GPL v2 para el núcleo). Ese mismo mes se abre Android Market, para la descarga de aplicaciones. En abril de 2009, Google lanza la versión 1.5 del SDK, que incorpora nuevas características como el teclado en pantalla. A finales de 2009 se lanza la versión 2.0 y a lo largo de 2010, las versiones 2.1, 2.2 y 2.3.

Durante el año 2010, Android se consolida como uno de los sistemas operativos para móviles más utilizados, con resultados cercanos a iOS e incluso superando al sistema de Apple en EE.UU.

Figura 3.1 Nombre y versiones de Android a lo largo de su historia.

En el año 2011 se lanza la versión 3.x (Honeycomb), específica para tabletas, y la 4.0 (Ice Cream Sandwich), tanto para móviles como para tabletas. Durante ese año, Android se consolida como la plataforma para móviles más importante y alcanza una cuota de mercado superior al 50 %.

En 2012, Google cambia su estrategia en su tienda de descargas *online*, reemplazando Android Market por Google Play Store, donde en un solo portal unifica tanto la descarga de aplicaciones como la de contenidos. Ese año aparecen las versiones 4.1 y 4.2 (Jelly Bean). Android mantiene su espectacular crecimiento y alcanza, a finales de año, una cuota de mercado del 70 %.

En 2013 se lanzan las versiones 4.3 y 4.4 (KitKat). En 2014 se lanza la versión 5.0 (Lollipop). A finales de ese año, la cuota de mercado de Android llega al 85 %. En octubre de 2015 ha aparecido la versión 6.0, con el nombre de Marshmallow. En 2016 se lanzó la versión 7.0 Android Nougat.

Android Oreo es el nombre de la última versión del sistema operativo móvil Android que anunció la firma Google el día 21 de marzo de 2017. Su nombre fue revelado el día 21 de agosto de 2017, el día del eclipse total de Sol en Estados Unidos.

3.1.3. Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Figura 3.2 Logotipo del software Android Studio.

Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android.

Estuvo en etapa de vista previa de acceso temprano a partir de la versión 0.1, en mayo de 2013, y luego entró en etapa beta a partir de la versión 0.8, lanzada en junio de 2014. La primera compilación estable, la versión 1.0, fue lanzada en diciembre de 2014. La última versión estable es la 3.0, y fue lanzada en octubre de 2017.

Android Studio está disponible para Windows 2003, Vista, 7, 8, y 10 tanto plataformas de 32 como de 64 bits, GNU/Linux, Linux con GNOME o KDE y 2 GB de memoria RAM mínimo y macOS, desde 10.8.5 en adelante.

Se espera que se desarrollen nuevas funciones con cada versión de Android Studio. Las siguientes características se proporcionan en la versión estable actual:

Tabla 3.2 Características del software Android Studio.

Características	**Funcionamiento**
Consola de desarrollador.	Consejos de optimización, ayuda para la traducción, estadísticas de uso.
Soporte para construcción	Basada en Gradle.
Editor de diseño enriquecido.	Permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario.
Herramientas Lint.	Detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
Plantillas	Crear diseños comunes de Android y otros componentes.
Soporte integrado para Google Cloud Platform	Permite la integración con Google Cloud Messaging y App Engine.
Dispositivo virtual de Android	Para ejecutar y probar aplicaciones.

Características
Integración de ProGuard y funciones de firma de aplicaciones.
Renderizado en tiempo real.
Refactorización específica de Android y arreglos rápidos.
Soporte para programar aplicaciones para Android Wear.

3.2. Reconocimiento de Voz

El reconocimiento de voz es la capacidad de un ordenador, de convertir, las palabras de la voz humana a un código binario comprensible por la computadora. La mayoría de las personas tienen la idea de que el reconocimiento de voz, se basa en que un computador tiene una especie de oídos electrónico, en realidad este sirve más como un traductor, el cual convierte nuestro lenguaje, en uno comprensible por la máquina. Este tipo de funciones mejoran las experiencias del usuario.

El reconocimiento de voz funciona en muchos niveles. Como una derivación del concepto, el reconocimiento de voz, es el intento del equipo para identificar a la persona que le habla, basándose en el tono único de su voz.

El proceso de la identificación de personas a través del reconocimiento de voz depende de diversas características del individuo: por un lado, está la estructura física del tracto vocal; por otro se encuentran ciertas características de comportamiento. En el momento del proceso de identificación se ha de tener muy en cuenta la variabilidad que posee la señal de voz, pues el individuo no puede repetir de forma completamente exacta una misma palabra o frase.

3.2.1. Modelo Acústico de la Voz

Tradicionalmente necesitamos de tres grupos o sistemas de órganos para que, coordinadamente podamos producir lenguaje hablado. Estos grupos son: Sistema Respiratorio, Sistema Fonatorio y Sistema Articulatorio. Mediante estos sistemas se puede definir un modelo acústico de generación de voz humana, el cual considera las siguientes etapas:

- Generación del sonido madre: Mediante vibración de las cuerdas vocales (se distinguen dentro de esta categoría al menos tres patrones vibratorios diferentes), mediante flujo de aire y mediante golpes de presión acumulada.
- Modulación del tracto vocal: Se puede considerar al tracto vocal como tubo de sección variable, controlado según la articulación determinada por los músculos de la faringe, el cuerpo y punta de la lengua, el velo del paladar y los labios. Al igual que en un tubo simple, el fenómeno de resonancias en el tracto aumenta la amplitud de un grupo de frecuencias alrededor de una determinada banda de frecuencia. A cada resonancia se le denomina formante.
- Radiación de salida: La salida del sonido tiene asociada una impedancia de radiación que influye sobre el sonido variando la composición espectral del sonido y los niveles de presión sonora con respecto al ángulo de salida.

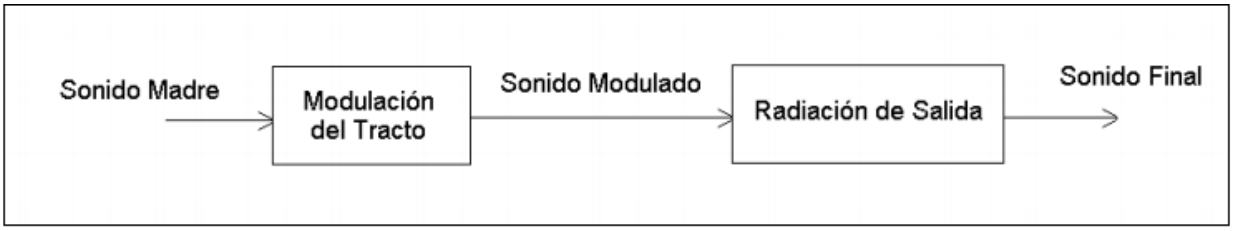

Figura 3.3 Modelo acústico básico de la generación de voz.

Una parte fundamental dentro del reconocimiento de voz son los fonemas. Un fonema es la mínima unidad lingüística capaz de producir cambios de significados. Estos fonemas pueden ser agrupados en dos grandes entidades: Fonemas Vocálicos y Fonemas Consonánticos. Ambos tipos de fonemas pueden clasificarse según el estado vibratorio de las cuerdas vocales, en fonemas fonados (ó sonoros) y áfonos (ó sordos). Los Fonemas Vocálicos son siempre fonados, a diferencia de los consonánticos que pueden ser o no fonados. Sin embargo, la principal diferencia entre los fonemas vocálicos y consonánticos es el tamaño de estas constricciones asociadas a las articulaciones en el tracto, siendo en las consonantes estas obstrucciones mucho más estrechas. Las vocales se clasifican según la apertura del tracto vocal (vocales abiertas y cerradas) y según el grado de elevación del dorso de la lengua (vocales anteriores, centrales y posteriores). Por otro lado las consonantes se clasifican según el punto de articulación (lugar de contacto de los articuladores que participan en la producción de un fonema específico), modo de articulación (forma como se interrumpe el flujo aéreo espiratorio), función de las cuerdas vocales (presencia o ausencia de este al momento de producir un fonema), y posición del velo del paladar (capacidad de controlar el paso de aire hacia la cavidad nasal u oral).

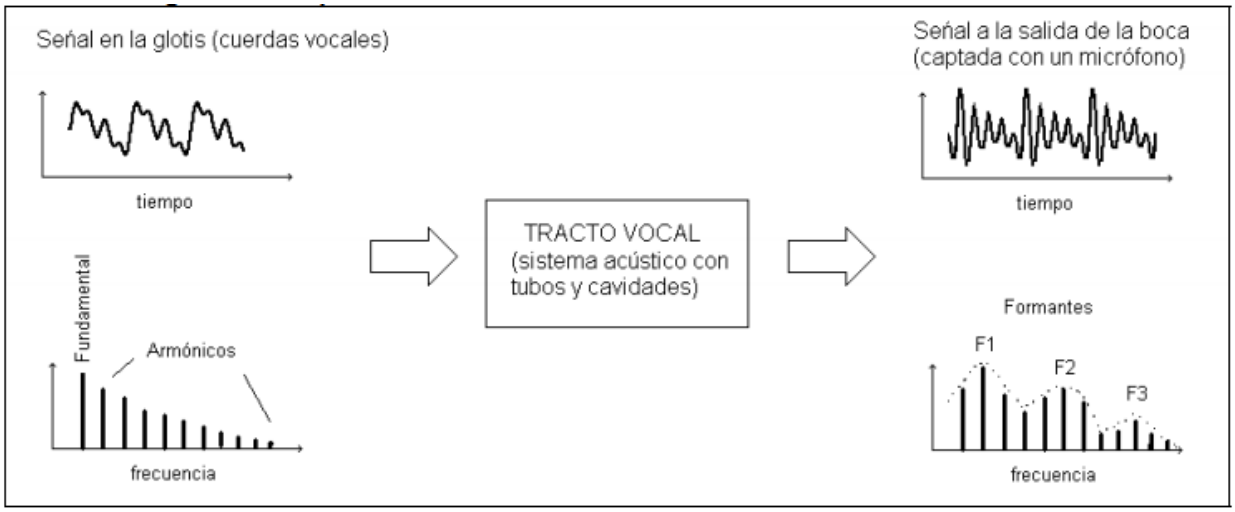

Figura 3.4 Esquema del efecto del tracto vocal sobre la señal del sonido madre.

3.2.2. Procesamiento de la Señal de Voz

Convertir la entrada de voz a una forma que el reconocedor pueda procesar o que la señal sea más accesible para procesar luego. Se concentra en el análisis y en el procesamiento de señales representadas en forma digital, es decir, discretizadas en el tiempo y en la amplitud".

A su vez, "Con las técnicas digitales, se produce una taza de error extremadamente baja, produciendo una señal de alta fidelidad con posibilidad de detección de error y corrección por un proceso similar que no es compatible con los analógicos".

Es de suma importancia considerar la discretización en el tiempo, la discretización en amplitud y la codificación o digitalización de la señal, más aun si la señal eléctrica a tratar es de tipo analógica; la primera se define como el muestreo de la señal y matemáticamente modifica las ecuaciones de las trasformadas, convolución, correlación, entre otras y si no cumple con ciertas premisas como el teorema de Nyquist genera un posible problema, denominado *aliasing* que no es más que el solapamiento de la señal y se origina cuando la tasa de muestreo es insuficiente, ocasionando una perdida irrecuperable de la información contenida en la señal.

Por otra parte, la segunda consideración es definida como la cuantificación (cuantización) de la señal, esta parte del proceso puede ser casi imperceptible si se cuantifica la señal con muchos bits o muy notoria si la cuantificación consta de pocos bits, la discretización en amplitud puede provocar algunos efectos indeseables, tales como: Si procede de la conversión A/D de la señal, adiciona un fenómeno denominado ruido de cuantización, también afecta los cálculos y si es significativo, puede producir errores importantes e incluso inestabilidad en algunos sistemas.

Finalmente, la codificación consiste en digitalizar la señal en valores binarios (0 y 1), representados de acuerdo al número de símbolos o bits y para ello existen varias técnicas de codificación, tales como; NRZ, UNRZ, mánchester, entre otras.

De esta manera se aprecia que el PDS requiere en su sistema de un filtro *antialiasing*, de un convertidor A/D (muestreo, cuantificación y codificación de la señal), un procesador DSP, un convertidor D/A y un filtro analógico para suavizar la salida, tal y como lo muestra la siguiente ilustración.

Ilustrando la señal de entrada analógica y los pasos del procesamiento digital de señales (PDS), el primer bloque representa un filtro pasabajo análogo, encargado de limpiar la señal antes de realizar el muestreo; el argumento teórico de esta necesidad de filtrado viene dada por el teorema de Nyquist, el cual indico "el efecto producido en el espectro de la frecuencia de una señal analógica al ser discretizada en el tiempo" y por Claude Shanon, quien demostró que "es posible reconstruir perfectamente una señal analógica a partir de sus muestras, si se dispone de un filtro pasabajos análogos".

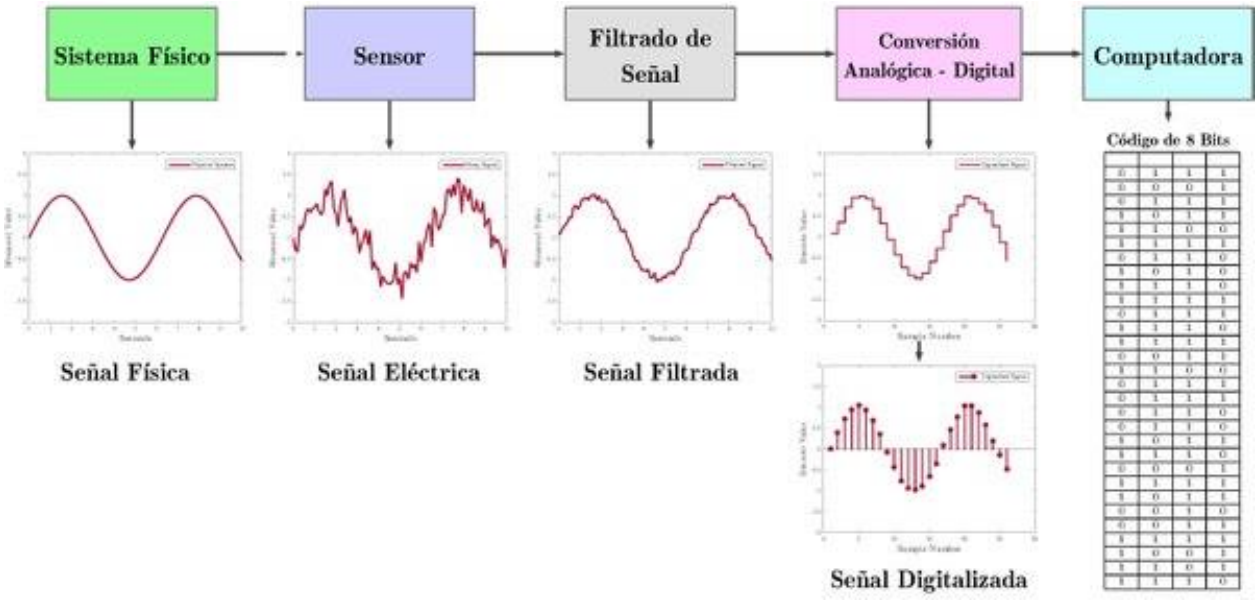

Figura 3.5 Sistema de procesamiento digital de señales

Luego la segunda sección en la figura representa la conversión analógica-digital que no es más que el muestreo, la cuantificación y la codificación de la señal. Posteriormente el bloque PDS, es similar a un procesador o microprocesador, que tiene la finalidad del procesado digital, ajustados a las necesidades requeridas y a la aplicación que se le otorgue.

3.2.3 Autocorrelación y determinación del Pitch

La autocorrelación o dependencia secuencial es una herramienta estadística utilizada frecuentemente en el procesado de señales.

La función de autocorrelación se define como la correlación cruzada de la señal consigo misma. Resulta de gran utilidad para encontrar patrones repetitivos dentro de una señal, como la periodicidad de una señal enmascarada bajo el ruido o para identificar la frecuencia fundamental de una señal que no contiene dicha componente, pero aparecen numerosas frecuencias armónicas de esta.

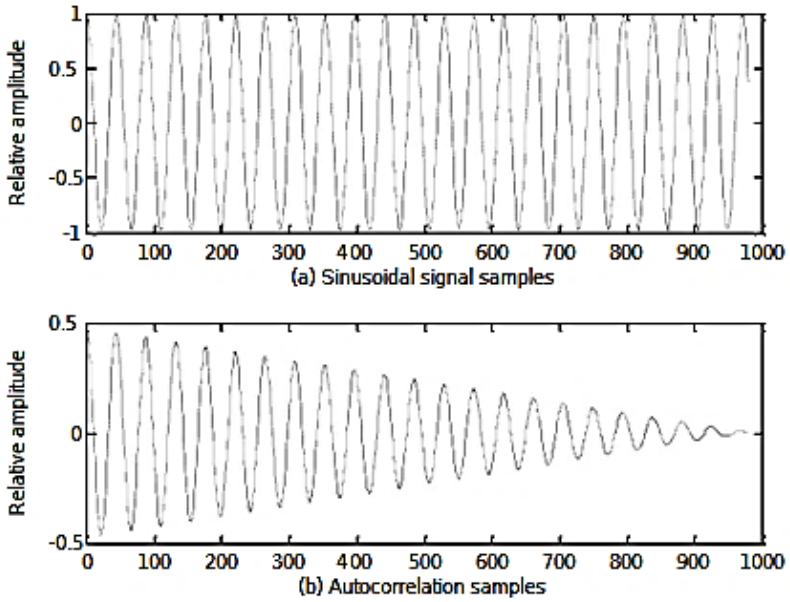

Figura 3.6 Función autocorrelación de una señal sinusoidal.

La función de autocorrelación general es conocida como ACF y está definida como:

$$\Phi(k) = \sum_{m=-\infty}^{\infty} s(m)\, s(m+k) \qquad (3.1)$$

El ACF es grande cuando una versión retardada de la señal es similar a la señal (es decir, correlacionada), donde k es igual al desfase (el ACF es máximo, cuando en k es igual 0 y es equivalente a la energía). Así, ACF es grande en retardos iguales a periodicidades contenidas en la señal. Por ejemplo, si la señal s es periódica con periodo P, entonces $\Phi(0) = \Phi(P) = \Phi(2P)$ ...

Para calcular el plazo corto de ACF, se requiere de ventanas como:

$$R_n(k) = \sum_{m=-\infty}^{\infty} [s(m)w(n-m)][s(m+k)w(n-k-m)] \qquad (3.2)$$

Considerando que en un principio Rn(0) = En siendo éste, la energía de corta duración. Se puede observar en la voz sonora el ACF se tiene un pico k igual al siguiente pico de la voz (así como en los múltiplos del tono), mientras que en la voz sorda es evidente la poca correlación que existe.

En caso de la voz sonora, para el segmento de la voz el espacio puede ser estimado por el retraso del valor máximo de la ACF en algún rango válido (por ejemplo, 60 a 350Hz, que es rango de frecuencias para la localización del Pitch) o, de manera similar, por desfase del mínimo de la AMDF.

A veces, es útil retirar primero la parte de la correlación de muestra a muestra (el primer pico del segmento analizado) por una operación de recorte de centro. Este tipo de métodos que utilizan la detección de picos o formantes de la señal de voz han tenido un cierto éxito.

Existen técnicas de procesamiento de señales de voz más complejos y robustos que también pueden ser utilizados para la detección del tono, como lo es la predicción lineal, que no son de nuestro mayor interés, pero serán explicadas brevemente, para su mayor entendimiento.

3.3. Sistemas de Encriptación

El objeto de la Criptografía es permitir la trasmisión de información privada por un canal inseguro, de forma que cualquier intruso que intercepte la comunicación no entienda su significado. Hasta hace unos treinta años, a la criptografía era casi exclusiva de gobiernos y mandos militares, los únicos que necesitaban proteger sus comunicaciones

3.3.1. Antecedentes

Los historiadores dicen que la criptografía es casi tan antigua como la propia escritura. Afirman que está presente en todas las civilizaciones de la antigüedad y dan ejemplos documentados que lo demuestran. El uso regular de la criptografía comienza en la edad media con los árabes, y en Europa ello no sucede hasta el Renacimiento.

El primer texto relacionado con la criptografía del que se tiene conocimiento data aproximadamente del año 1900 a.C. Es del antiguo Egipto: un grabado en piedra de la cámara principal de la tumba de un noble de la ciudad de Menet Khufu, a las orillas del Nilo. En realidad, no es un texto criptografiado con la intención de ocultar su contenido, sino que es un simplemente un texto en el que ciertos símbolos jeroglíficos se cambian por otros similares, pero no usuales.

Los escribas de la antigua Mesopotamia también cambiaron en ocasiones los signos cuneiformes de su escritura por otros, coincidiendo así con sus colegas egipcios en esta forma de alterar la escritura. Pero a diferencia de los egipcios, los escribas mesopotámicos si tuvieron la intención de ocultar el significado de la escritura. De esta cultura es el texto cifrado mas antiguo que se conserva, data aproximadamente del año 1500 a.C. Es una tablilla de arcilla en la que se escribió secretamente una formula para el barniz que se empleaba en alfarería. Seguramente era un valioso tesoro en aquella remota época.

Saltando ya al siglo VI a.C., en algunos antiguos textos hebreos, entre los que están los bíblicos, figuran nombres de personas y ciudades que han sido transformados mediante ciertas sustituciones de unas letras por otras. La mas frecuente es la denominada *atbash.* En el *atbash* la primera letra del alfabeto hebreo se cambia por la ultima y viceversa, la segunda por la antepenúltima y así sucesivamente, según el esquema que se muestra en la figura **XX**.

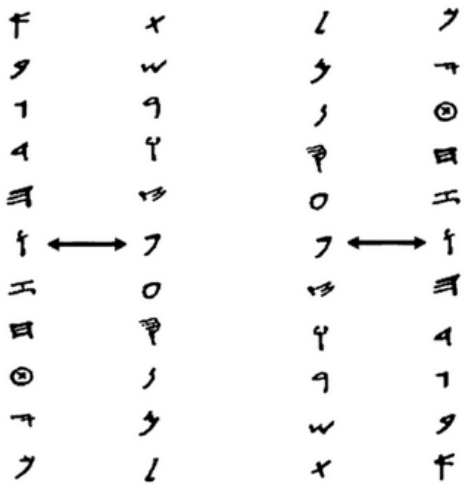

Figura 3.7 El "atbash" hebreo.

Unos pocos años después, en Roma se desarrolló el cifrador de César, que recibía este nombre en honor al emperador romano Julio César; desde luego el César no lo inventó, pero si lo utilizo para enviar y recibir mensajes en las múltiples guerras que se libraron durante las conquistas romanas. Su funcionamiento es muy sencillo. Consiste en escribir las letras del alfabeto en un renglón; luego en el renglón inmediato inferior se recorre la escritura del alfabeto tres lugares, tal como muestra la figura **XX**. Como este cifrado utiliza un solo alfabeto, se dice que emplea la técnica de sustitución mono alfabética.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z			
			a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z

Figura 3.8 Cifrador del César.

Durante la Primera Guerra Mundial, la criptografía se usó de manera intensiva. Alemania desarrolló el código Ubchi que fue desarmado por Francia y los códigos navales de Alemania fueron descifrados por el " Room 40" del Almirantazgo de Reino Unido, lo cual les permitió adelantarse a Alemania y prepararse para batallas como la de Jutlandia.

Criptografía en la Segunda Guerra Mundial

La criptografía fue clave durante la Segunda Guerra Mundial y, de hecho, hizo cambiar el curso de la guerra. Alemania había conseguido dominar el Atlántico Norte con su flota de submarinos, y sus comunicaciones eran indescifrables gracias a la máquina Enigma. Además de los frentes tradicionales y las batallas entre las fuerzas armadas se había abierto un nuevo campo de batalla: romper las comunicaciones enemigas; una tarea que los Aliados encargaron a un grupo de matemáticos, ingenieros y físicos que trabajaron desde las instalaciones de Bletchley Park y entre los que se encontraba Alan Turing.

Quizás el trabajo de Alan Turing y los Aliados sea la labor más conocida sobre criptografía durante la Segunda Guerra Mundial; sin embargo no fue el único. El cifrado de las comunicaciones marcó el conflicto y se empleó un conjunto muy variado de técnicas para evitar que el enemigo interceptase las comunicaciones. Estados Unidos, por ejemplo, rescató una técnica que ya había utilizado con éxito durante la Primera Guerra Mundial y, en vez de recurrir a complejos algoritmos de cifrado, apostó por usar como código los idiomas de los nativos americanos.

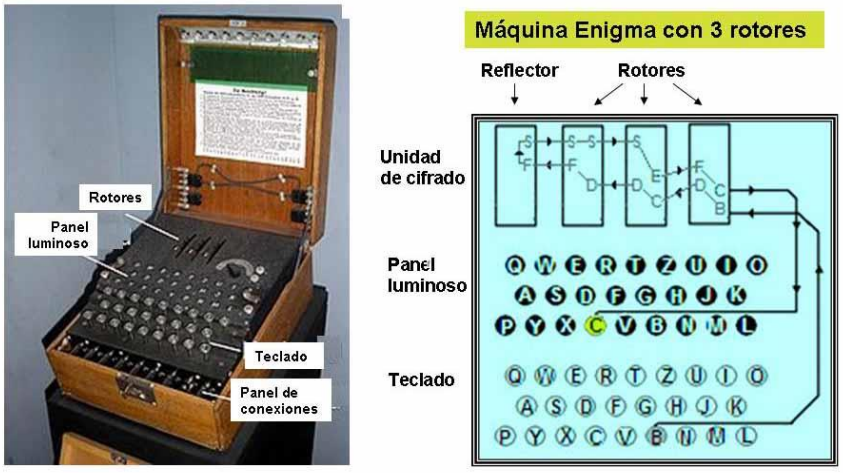

Figura 3.9 Máquina Enigma.

También en el ámbito del frente del Pacífico, el esfuerzo por romper las claves de cifrado que usaba Japón era clave para detener su avance. Gracias al esfuerzo conjunto de Estados Unidos y fuerzas de Holanda y Gran Bretaña, se consiguió romper el código naval japonés JN-25 y, de esta forma, conocer los planes de batalla de Japón en Midway.

La criptografía moderna y la era digital

Después de la Segunda Guerra Mundial, la criptografía dio un gran salto gracias a Claude Shannon, conocido como el padre de la teoría de la comunicación. En 1948, Shannon, que trabajaba en los Laboratorios Bell, publicó *A Communications Theory of Secrecy Systems*; un artículo fundamental en el que se modernizaron las técnicas de codificación para transformarlas en procesos matemáticos avanzados.

Si bien es cierto que el análisis de frecuencia se basaba en la estadística, Shannon demostró matemáticamente este hecho e introdujo el concepto de " distancia de unicidad" que marcaba la longitud de un texto cifrado que se necesita para poder descifrarlo.

Hasta el 17 de marzo de 1975 no llegaría el primer avance público (no dependiente de la NSA) vinculado al mundo de la criptografía. IBM desarrolló el algoritmo de cifrado Data Encryption Standard (DES) que, dos años más tarde, se convertiría en un Federal Information Processing Standard (FIPS 46-3) y se extendería su uso por todo el mundo. En el año 2001, DES cedería su puesto a Advanced Encryption Standard (AES) que, tras 5 años de revisión, se convirtió en un estándar.

Prácticamente, todos los sistemas de los que hemos hablado son simétricos; tanto emisor como receptor deben manejar el mismo código y estar informados mutuamente del código que van a usar a la hora de intercambiar información. Sin embargo, Whitfield Diffie y Martin Hellman sentaron las bases de la criptografía asimétrica (clave pública y clave privada) en el artículo "New Directions in Cryptography" publicado en 1976. La criptografía asimétrica hoy es fundamental para transacciones realizadas a través de Internet, por ejemplo, en páginas que usan el protocolo HTTPS o para cifrar nuestros mensajes usando PGP (que combina tanto criptografía asimétrica como criptografía asimétrica).

3.3.2. DES (Data Encryption Standard)

El 15 de mayo de 1973, el NBS (*National Bureau of Standars, en castellano: Agencia Nacional de Normalización*) hoy en día denominada *NIST* (*National Institute of Standars and Technology, en castellano: Instituto Nacional de Normalización y Tecnología*), hizo un llamamiento en el *Federal Register (el equivalente en España del Boletín Oficial del Estado)* para la creación de un algoritmo de cifrado que cumpliera con los siguientes requisitos:

- ofrecer un alto nivel de seguridad relacionado con una pequeña clave utilizada para cifrado y descifrado
- ser comprensible
- no depender de la confidencialidad del algoritmo
- ser adaptable y económico
- ser eficaz y exportable

A finales de 1974, IBM propuso "Lucifer", que gracias a la NSA (National Standard Agency, en castellano: Agencia Nacional de Seguridad) fue modificado el 23 de noviembre de 1976, convirtiéndose en DES (*Data Encryption Standard, en castellano: Estándar de Cifrado de Datos*). El DES fue aprobado por el NBS en 1978. El DES fue estandarizado por el *ANSI* (*American National Standard Institute, en castellano: Instituto Nacional Americano de Normalización*) bajo el nombre de *ANSI X3.92*, más conocido como *DEA* (*Data Encrytion Algorithm, en castellano: Algoritmo de Cifrado de Datos*).

Se trata de un sistema de cifrado simétrico por bloques de 64 bits, de los que 8 bits (un byte) se utilizan como control de paridad (para la verificación de la integridad de la clave). Cada uno de los bits de la clave de paridad (1 cada 8 bits) se utiliza para controlar uno de los bytes de la clave por paridad impar, es decir, que cada uno de los bits de paridad se ajusta para que tenga un número impar de "1" dentro del byte al que pertenece. Por lo tanto, la clave tiene una longitud "útil" de 56 bits, es decir, realmente sólo se utilizan 56 bits en el algoritmo.

El algoritmo se encarga de realizar combinaciones, sustituciones y permutaciones entre el texto a cifrar y la clave, asegurándose al mismo tiempo de que las operaciones puedan realizarse en ambas direcciones (para el descifrado). La combinación entre sustituciones y permutaciones se llama cifrado del producto.

La clave es codificada en 64 bits y se compone de 16 bloques de 4 bits, generalmente anotadas dek_1 a k_{16}. Dado que "solamente" 56 bits sirven para el cifrado, puede haber hasta 2^{56} (o $7.2*10^{16}$) claves diferentes.

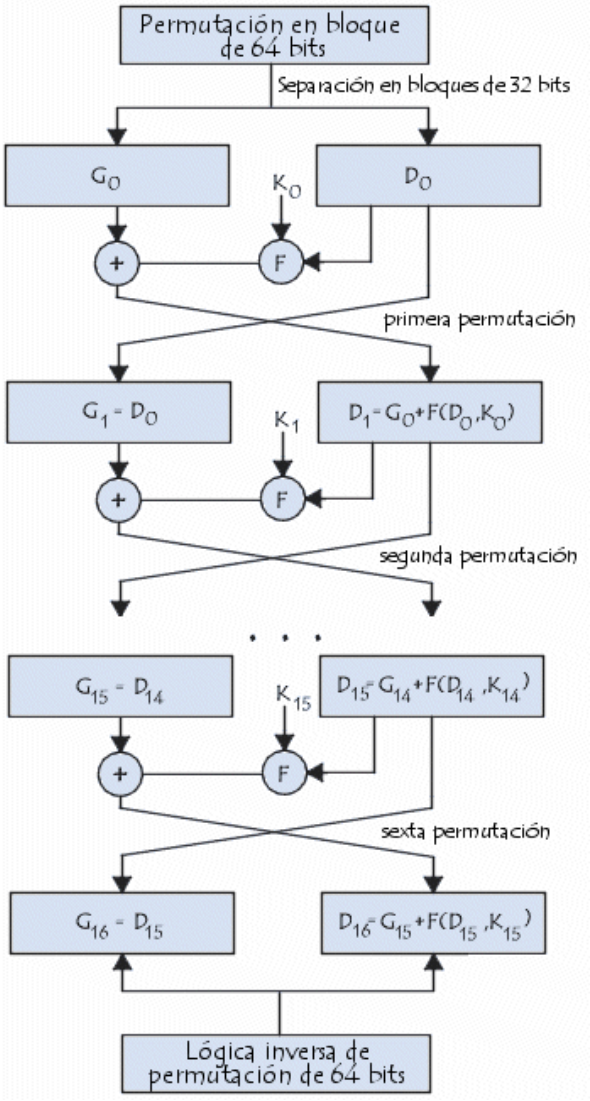

Figura 3.10 Principio de funcionamiento DES.

Las partes principales del algoritmo son las siguientes:

- fraccionamiento del texto en bloques de 64 bits (8 bytes),
- permutación inicial de los bloques,
- partición de los bloques en dos partes: izquierda y derecha, denominadas *I* y *D*respectivamente,
- fases de permutación y de sustitución repetidas 16 veces (denominadas rondas), reconexión de las partes izquierda y derecha, seguida de la permutación inicial inversa.

3.3.3. RC4 (Rivest Cipher 4)

El algoritmo RC4 fue diseñado en 1987 por Ron Rivest de la empresa RSA Security. El nombre RC4 es un acrónimo de Rivest Cipher 4, aunque también se dice que su significado es el de Ron´s Code 4. Este algoritmo se basa en generar una clave de forma pseudoaleatoria que tiene la misma longitud que el texto original. A esta clave y al texto original se le aplica la operación lógica XOR (O exclusiva), dando como resultado un texto cifrado, que se muestra en la figura XX.

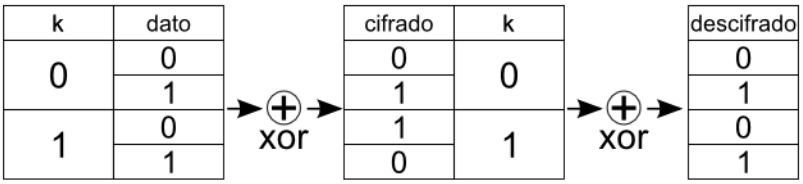

Figura 3.11 Cifrado de bits por medio de XOR.

RC4 es parte de los protocolos de cifrado más comunes como WEP, WPA para tarjetas wireless y TLS. Entre los factores principales que han ayudado a que RC4 esté en un rango tan amplio de aplicaciones son su increíble velocidad y simplicidad. La implementación tanto en software como en hardware es muy sencilla de desarrollar y son muy pocos los recursos necesarios para obtener un rendimiento eficiente.

El algoritmo de cifrado *RC4* se descompone en tres partes:

- Inicialización del vector de estado (*KSA, key-scheduling algorithm*).
- Generación del flujo de cifrado (*PRGA, pseudo-random generation algorithm*)
- Mezcla del texto con el flujo de cifrado.

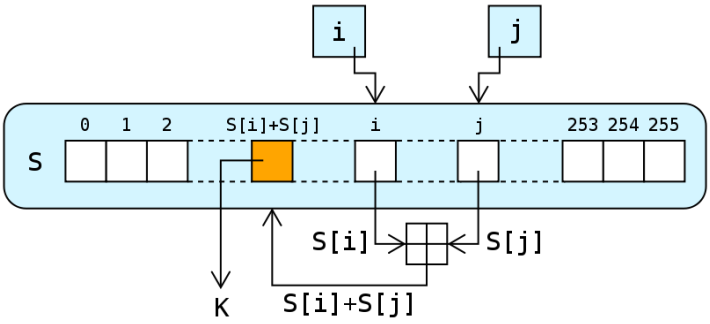

Figura 3.12 Diagrama del algoritmo RC4.

Inicialización del vector de estado

Para generar el flujo de cifrado, *RC4* utiliza una permutación de todos los bytes posibles — es decir, un vector *S* de 256 elementos con todos los números del 0 al 255.Este vector se utilizará como entrada en la siguiente etapa del algoritmo para generar el flujo de cifrado.

Emisor y receptor comparten una clave secreta. El algoritmo *RC4* admite claves de hasta 256 bytes de longitud, aunque la máxima seguridad se consigue con tamaños significativamente menores. Típicamente se utilizan claves de entre 5 y 16 bytes (40 - 128 bits).

El *RC4* contempla un protocolo para desordenar los 256 elementos del vector a partir de la clave secreta compartida por emisor y receptor. En este protocolo, se carga el vector *S* con valores ordenados del 0 al 255 y se itera por cada elemento, intercambiando cada uno por otro de los elementos del vector determinado por la combinación de uno de los números (bytes) de la clave y el valor actual del elemento.

En el cálculo de la posición de intercambio para cada iteración se tiene en cuenta el resultado de la iteración anterior, haciendo el algoritmo más difícil de atacar. El resultado es un vector permutación *S* de aspecto aleatorio, pero replicable fácilmente por cualquiera que tenga la clave secreta (receptor legítimo).

Generación del flujo de cifrado

Partiendo del vector de estado generado en la etapa anterior y dos variables *i* y *j* se genera el flujo de cifrado. En cada iteración del algoritmo se emite un byte del flujo de cifrado y se intercambian dos elementos del vector. En todo momento el estado del sistema está determinado por el valor de las variables *i* y *j* y el vector estado.

Si un atacante llegara a conocer todos estos valores en un momento determinado de la transmisión, podría descifrar todos los mensajes que se cifraran desde ese momento en adelante.

Mezcla del texto con el flujo de cifrado (*keystream*)

A cada *K* resultado del algoritmo anterior se le hace la operación *XOR* con un byte del texto plano para obtener el texto cifrado y se envía por el canal.

CAPÍTULO 4. Diseño e Implementación

4.1. Diseño y desarrollo del sistema

En este capítulo se muestra el diseño y desarrollo de la aplicación, la cual consiste en una interfaz gráfica de usuario, que utiliza lenguaje XML para la parte visual y lenguaje Java en la parte operativa, en la que el usuario accede a la aplicación por medio de reconocimiento de voz. El reconocimiento de voz se realiza usando el principio de autocorrelación y la frecuencia fundamental.

Al ser reconocido el usuario podrá acceder a archivos los cuales será capaz de encriptar y desencriptar por medio de un algoritmo tipo RC4.

Primeramente, el usuario debe registrarse; para ello se realizan 3 grabaciones por medio de la grabadora de voz de Android, la cual hace uso del micrófono interno del celular. Estas grabaciones quedan almacenadas en la memoria interna del celular y son llamadas a una función que hace el uso de autocorrelación, definida por la fórmula 4.1, para encontrar los patrones repetitivos dentro de las grabaciones y así poder identificar la frecuencia fundamental de cada una, cuya definición está dada por la fórmula 4.2,. Posteriormente se realiza un promedio de los tres valores de frecuencia fundamental obtenidos y se almacena para su comparación para el acceso del usuario.

$$r_{xx}(l) = \frac{1}{N}\sum_{n=0}^{N-1} x(n+l)x(n) \qquad (4.1)$$

Donde N = tamaño de la ventana de muestras
l = desplazamiento o retraso
n = índice de la sumatoria
x( ) = señal original

$$f_{xx}(l) = \frac{f_s}{r_{xx}(l)} \qquad (4.2)$$

Donde l = desplazamiento o retraso
fs = frecuencia de muestreo
r( ) = autocorrelación de la señal

Finalmente, el usuario debe realizar una última grabación la cual llevara el mismo procesamiento para calcular su frecuencia fundamental. Una vez calculado, este valor se compara con el promedio almacenado en la sección de registro. Si la diferencia entre ambas es menor a un valor mínimo propuesto, el usuario podrá acceder; de lo contrario su solicitud será denegada.

Después de ser validado el usuario, se le muestran dos tablas. Una contiene los archivos previamente guardados en una carpeta designada por la aplicación que se desean encriptar; la segunda tabla muestra los archivos que ya han sido encriptados y que el usuario puede desencriptar para poder ver su contenido.

4.2. Diseño del Software

En el proyecto se requiere de tres fases:

- Registro: se requiere de la grabación, procesamiento y almacenamiento de las tres muestras de audio, de las cuales se obtiene su frecuencia fundamental y es calculado su promedio, el cual es almacenado para compararlo posteriormente para permitir el acceso al usuario.
- Reconocimiento: al terminar su registro, se pide al usuario una última grabación requiriendo que diga su contraseña previamente almacenada. Ésta es procesada y comparada con el promedio obtenido en el registro. La diferencia entre ambos debe ser menor al valor mínimo establecido para que el usuario pueda entrar (valor mínimo=20Hz).
- Encriptación: se requiere de la lectura de los archivos que se desean encriptar/desencriptar de manera binaria y hacer uso de las funciones dentro de la librería Java para seleccionar el tipo de algoritmo de encriptación que se desea ocupar, en nuestro caso RC4. En el caso de desencriptacion se utiliza la misma metodología, solo cambia los argumentos de la función por la fase de desencriptar.

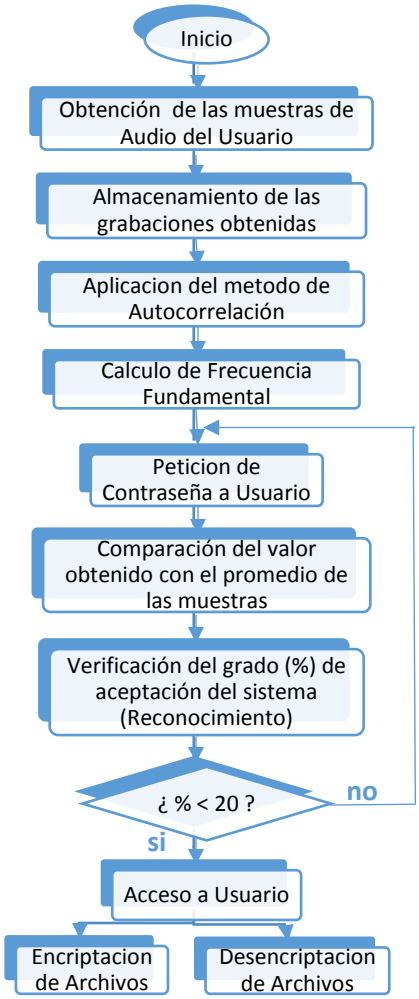

Figura 4.1 Diagrama general del sistema.

4.2.1. Interfaz Gráfica de Usuario

El empleo de una interfaz gráfica de usuario permite de un modo más interactivo la visualización del entorno. Para desarrollar la parte visual se empleó el lenguaje XML que Android Studio utiliza para el modelado de elementos visuales.

La aplicación se divide en tres interfaces (*activity*), donde cada una cumple una de las tres funciones mencionadas anteriormente (registro, reconocimiento, encriptación).

Se cuenta con dos pantallas más que se despliegan al momento en que el usuario es validado, mostrando el mensaje de acceso permitido o denegado además de desplegar el valor de la frecuencia fundamental del usuario.

4.2.2. Funciones de la Interfaz Gráfica de Usuario

Menú Principal

En esta primera interfaz se da la bienvenida al usuario y se le brindas las instrucciones para hacer uso de la aplicación. Cuenta con 2 secciones: registro y acceso.

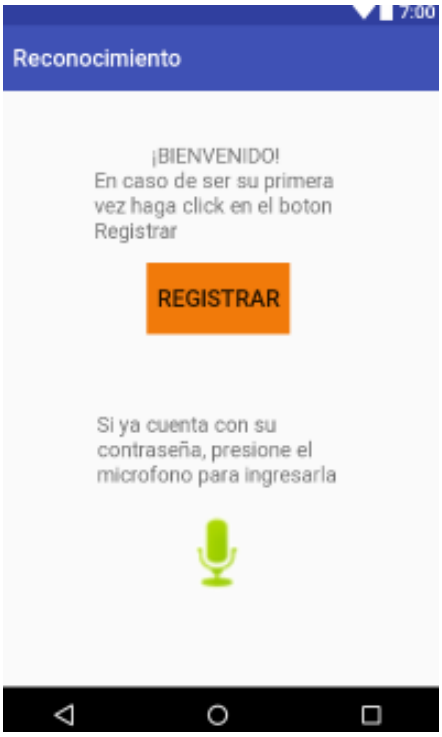

Figura 4.2 Menú principal

Declaración de variables.

Se hace uso de 4 variable, las dos primeras hacen referencia a los botones que el usuario podrá utilizar para interactuar con la interfaz; mientras que las 2 últimas son las que permiten la transición de una int6 a otra al terminar sus funciones.

```
public class MainActivity extends AppCompatActivity {
    ImageButton acceso;
    Button registro;
    int peticion=1;
    Uri url1;
```

Cuadro 4.1 Declaración de Variables

Vinculación de variables con los elementos gráficos.

Al crear las variables, éstas deben vincularse con los elementos gráficos de la interfaz, para ello se debe vincular con el nombre del elemento que se muestra en el archivo XML de la interfaz, en la parte superior de la barra de propiedades. La barra de propiedades aparece al lado derecho de la pantalla al dar click sobre el elemento de la sección de Diseño.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //CREACION DE BOTONES
        acceso = (ImageButton) findViewById(R.id.imageButton);
        registro = (Button) findViewById(R.id.button);
```

Cuadro 4.2 Vinculación de variables

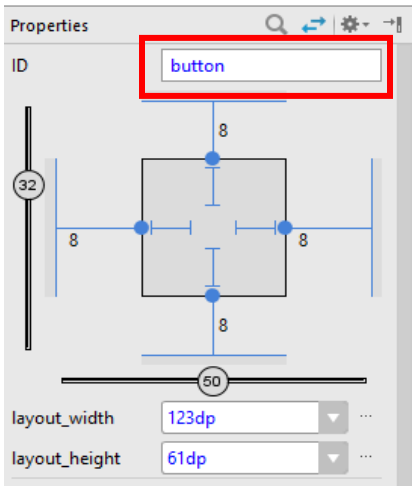

Figura 4.3 Barra de propiedades de Android Studio.

Botón Registro.

Haciendo uso de la función `setOnClickListener` se hace referencia a las funciones que se llevan a cabo cuando el usuario presiona el botón, en este caso el botón "Registrar". Al hacer click, se crea una variable `Intent` con la que cerramos la Activity actual y mandamos a llamar a la Activity2 para realizar la creación de la contraseña.

```
registro.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //Creamos el Intent
            Intent intent =
                    new Intent(MainActivity.this, Main2Activity.class);
            //INICIAMOS LA NUEVA ACTIVIDAD
            startActivity(intent);
        }
    });
```

Cuadro 4.3 Botón registro.

Botón de Acceso.

Al igual que el botón "Registrar", al hacer click se crea una variable `Intent` pero en este caso utilizamos la función `MediaStore`, la que nos permite hacer uso de aplicaciones internas originales del sistema operativo Android.
En nuestro caso, hemos usado la "Grabadora de Voz" de Android por medio del comando ***RECORD_SOUND_ACTION*** para que el usuario pueda crear y almacenar su contraseña por medio de un comando de voz.

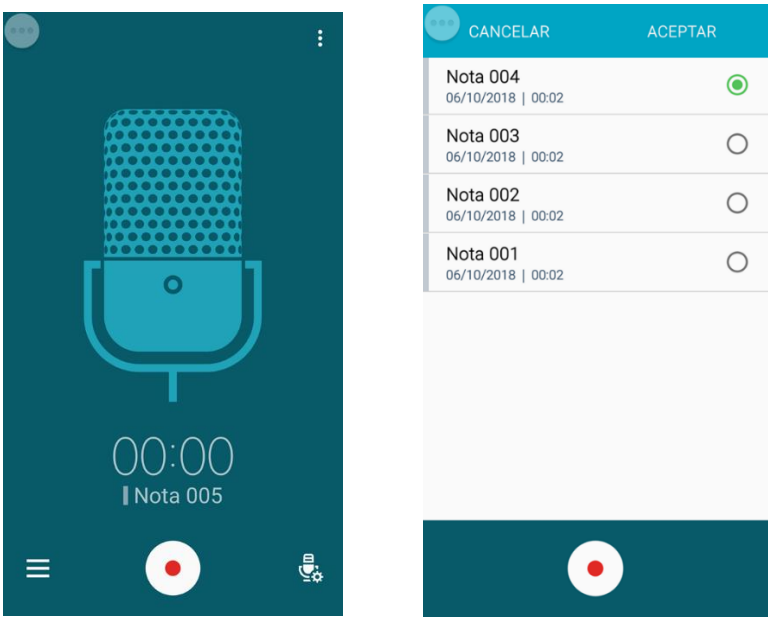

Figura 4.4 Grabadora de voz de Android.

```
acceso.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //Creamos el Intent
            Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
            startActivityForResult(intent, peticion);
        }
    });
}
```

Cuadro 4.4 Botón acceso.

Función del Botón de Acceso.

Al seleccionar el botón para poder accesar a la aplicación, se activa automáticamente la función `onActivityResult`, que lleva a la parte en que se leen las tres grabaciones adquiridas en el botón "Registro" y la grabación de acceso del usuario.

La función `leerFichero()` regresa el valor de la de la frecuencia fundamental de cada una. Con los valores obtenidos de las tres muestras se calcula su promedio, posteriormente se hace una diferencia entre el promedio y el valor de la frecuencia fundamental del audio de acceso.

Se aplica el valor absoluto al resultado para eliminar valores negativos, luego se pregunta si el valor obtenido es menor a un valor mínimo que se ha propuesto (en nuestro caso 20). Si la condición se cumple, el programa nos da acceso a la Activity3, en la que el usuario es capaz de encriptar o desencriptar sus archivos; en caso de que no se cumpla la condición, la aplicación e mantiene en la pantalla de inicio y manda un mensaje al usuario notificándole que no ha sido identificada su voz.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == peticion) {
        url1 = data.getData();
    }
    //Lectura de Audios y Calculo de Frec Fundamental
    float F1 = leerFichero("/storage/emulated/0/Sounds/Nota 001.m4a");
    float F2 = leerFichero("/storage/emulated/0/Sounds/Nota 002.m4a");
    float F3 = leerFichero("/storage/emulated/0/Sounds/Nota 003.m4a");
    float FU = leerFichero("/storage/emulated/0/Sounds/Nota 004.m4a");
    float Prom = (F1+F2+F3)/3;
    float RecUs = Math.abs(FU-Prom);
    //Reconocimiento
    if (RecUs<20){
        //Creamos el Intent
        Intent intent2 = new Intent(MainActivity.this, Main3Activity.class);
        //INICIAMOS LA NUEVA ACTIVIDAD
        startActivity(intent2);
    }else{
        Toast.makeText(getApplicationContext(), "Usuario no Identificado.
         Intente de Nuevo", Toast.LENGTH_LONG).show();
    }
}
```

Cuadro 4.5 Función del botón acceso.

Lectura de los Archivos de Audio y Autocorrelación.

Al ser llamada la función `leerFichero()`, se pide a la función mandar la ruta de donde se almacena las grabaciones, que en nuestro caso es **`/storage/emulated/0/Sounds`**. Al ser encontrado el archivo, se crean los objetos `FileInputStream` y `DataInputStream`, el primero crea un contenedor en el que se guardaran temporalmente los datos leídos del audio; el segundo nos da acceso a la función `read()` que es la encargada de leer los elementos del archivo de audio.

Posteriormente, se define un ciclo en el que se lee el archivo el cual contiene un contador (`tam`) para conocer el tamaño en bytes del archivo; se necesita conocer el tamaño el archivo para crear una variable del tamaño necesario para almacenar los datos y que no se desperdicie espacio de

memoria. Al terminar de leer se cierra el archivo y se crea un arreglo del tamaño de la grabación `Double[tam] datos` en que se almacenan los datos leídos para su procesamiento.

Se vuelve hacer uso de los objetos `FileInputStream` y `DataInputStream`, para abrir y leer nuevamente la grabación, creamos un ciclo para leer los datos y los almacenamos en cada localidad del arreglo `datos`.

Se cierra nuevamente los objetos, y se crea una variable de tipo double (`datos`) con la que recorreremos los datos del archivo y hacer un recorrido para encontrar el valor mas alto para cumplir con el principio de la autocorrelación.

Mientras se realiza el recorrido, se agrega una condición en la que cuente las posiciones para encontrar en cual se encuentra el dato con mayor valor.

Una vez encontrado, se aplica la formula 4.2 que se vio anteriormente para calcular la frecuencia fundamental, en la que dividimos la frecuencia de muestreo a la que graba nuestro dispositivo, que en este caso es una frecuencia de 44100 Hz, entre la posición en la que se encuentra nuestro valor máximo.

Finalmente, nuestra función regresa el valor obtenido y se almacena en las variables declaradas en la sección del "Botón de Acceso".

```
FileInputStream fis=null;
DataInputStream entrada=null;
int tam=0;
try {
    fis = new FileInputStream(Ruta);
    entrada = new DataInputStream(fis);
    while(entrada.readDouble() != 0) {
        tam++;
    }
    entrada.close();
    fis.close();
}
catch(Exception e){
    e.printStackTrace();
}

Double[] datos=new Double[tam];
int i=0;
try {
    fis = new FileInputStream(Ruta);
    entrada = new DataInputStream(fis);
    while(i<tam){
        datos[i]=entrada.readDouble();
        i++;
    }
    entrada.close();
    fis.close();
}
catch(Exception e){
    e.printStackTrace();
}
 //AUTOCORRELACION…
```

Cuadro 4.6 Lectura de los archivos de audio.

```
//AUTOCORRELACION
    double[] acf = new double[Tam];
    double suma = 0;
    for (int k = 0; k < Tam; k++) {
        suma = 0;
        for (int n = 0; n < Tam - k; n++) {
            suma = suma + (datos[n] * datos[n + k]);
        }
        acf[k] = suma / Tam;
    }

//CALCULO DE FREC FUNDAMENTAL
    int Rmin = 44100/350;
    int Rmax = 44100/50;
    for (int j=Rmin; j<Rmax ; j++)
    {
        if (acf[j] >= mayor) {
            mayor = acf[j];
            p=j;
        }
    }
    float ff = 44100/p;
    return ff;
}
```

Cuadro 4.7 Función de Autocorrelación y cálculo del Pitch.

Menú de Registro

En esta sección se hace referencia a los botones que el usuario podrá abrir la "Grabadora de Voz" de Android y poder registrar su contraseña. El botón siguiente es la que nos llevara de regreso a la pantalla de inicio una vez que el usuario haya terminado de registrar su contraseña.

Figura 4.5 Menú registro.

Botones para muestras de Audio.

Al hacer click a cada uno de los botones en forma de micrófono, se crea una variable Intent en el que utilizamos la función MediaStore, que por medio del comando *RECORD_SOUND_ACTION* se accesa a la "Grabadora de Voz" para que el usuario almacene 3 grabaciones (rec1, rec2, rec3), las cuales serán las muestras para obtener el parámetro con el que se compara la grabación de acceso que el usuario brinda para ser reconocido y poder entrar a la aplicación.

```
rec1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Creamos el Intent
        Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
        startActivityForResult(intent, peticion);
    }
});
```

Cuadro 4.8 Botón de grabación muestra.

Botón Siguiente.

Haciendo uso de la función `setOnClickListener` se hace referencia a las funciones que se llevan a cabo cuando el usuario presiona el botón, en este caso el botón "Siguiente". Al hacer click, se crea una variable `Intent` con la que cerramos la Activity2 y mandamos a llamar a la Activity1 que es la pantalla de inicio.

```
siguiente.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //Creamos el Intent
            Intent intent =
                    new Intent(Main2Activity.this, MainActivity.class);
            //INICIAMOS LA NUEVA ACTIVIDAD
            startActivity(intent);
        }
    });
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == peticion) {
        url1 = data.getData();
    }
}
```

Cuadro 4.9 Botón siguiente.

Interfaz de manejo de documentos

En esta sección hace uso principalmente de 2 elementos. En el caso de la variable tabs, el TabHost es el contenedor de las dos tablas que mostrara a los usuarios los archivos que has sido encriptados y los archivos que se encuentran en su forma original.

Se crea una variable tam se usara posteriormente para guardar el tamaño del arreglo byte que almacenarán cada audio. Salir es el botón de salida de la aplicación, el cual redirige a la página de inicio una vez el usuario haya terminado de encriptar o desencriptar los archivos que requiera.

Creación de Listas.

Se crea un `ArrayAdapter` el cual encapsulara la cadena que contiene el nombre de cada archivo y su ubicación de almacenamiento, el cual debe vincularse con los elementos List que están alojados dentro del TabHost. Existe un por cada `ArrayAdapter` Tab.

```
final ArrayAdapter<String> mAdapter1;
mAdapter1=new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1);
final ArrayAdapter<String> mAdapter2;
mAdapter2=new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1);
```

Cuadro 4.10 Creación de listas.

Búsqueda de archivos para encriptar o desencriptar.

Se crea un arreglo `File[]` que apunte a la carpeta Crypto previamente creada en la memoria externa, donde se encuentran los documentos que el usuario desean encriptar. Utilizando la función `listFiles()` se almacenan los archivos dentro de la carpeta Crypto en el arreglo File.
Una vez teniendo la ubicación de los archivos, primero se hace una búsqueda de los elementos almacenados para descartar los que sean carpetas y que solo se pueda visualizar los archivos. Posteriormente se hace la búsqueda de los elementos para separar los archivos originales de los encriptados, para esta búsqueda se toma como referencia la cadena ".cripto" la cual será la terminación de cada archivo que sea encriptado.

```
//Busqueda de archivos para encriptar
File miDir=new File ("/storage/extSdCard/Crypto");
    File[] files = miDir.listFiles();
    for (File file : files) {
        if (!file.isDirectory()) {
            if(file.getName().contains(".cripto")){
            }else{
                mAdapter2.add(file.getName()+"\n"+file.getAbsolutePath());
            }
        }
    }

//Busqueda de archivos para desencriptar
File miDirCrip=new File ("/storage/emulated/0");
    File[] files = miDirCrip.listFiles();
    for (File file : files) {
        if (!file.isDirectory()) {
            if(file.getName().contains(".cripto")){
                mAdapter1.add(file.getName()+"\n"+file.getAbsolutePath());
            }
        }
    }
```

Cuadro 4.11 Búsqueda de archivos para encriptar o desencriptar.

Impresión de Listas.

Una vez separados los archivos entre encriptados y no encriptados, se pasa a vinculación de las direcciones con los ListView con la función `setAdapter()`, finalmente se agregan a su respectiva tabla por medio de la función `addTab(spec1)`.

```
final ListView listView=(ListView) findViewById(R.id.listView);
listView.setAdapter(mAdapter1);
final ListView listView2=(ListView) findViewById(R.id.listView2);
listView2.setAdapter(mAdapter2);

TabHost.TabSpec spec1 = tabs.newTabSpec("Tab1");
spec1.setIndicator("Tab1");
spec1.setContent(R.id.listView);
tabs.addTab(spec1);

TabHost.TabSpec spec2 = tabs.newTabSpec("Tab2");
spec2.setIndicator("Tab2");
spec2.setContent(R.id.listView2);
tabs.addTab(spec2);
```

Cuadro 4.12 Impresión de listas.

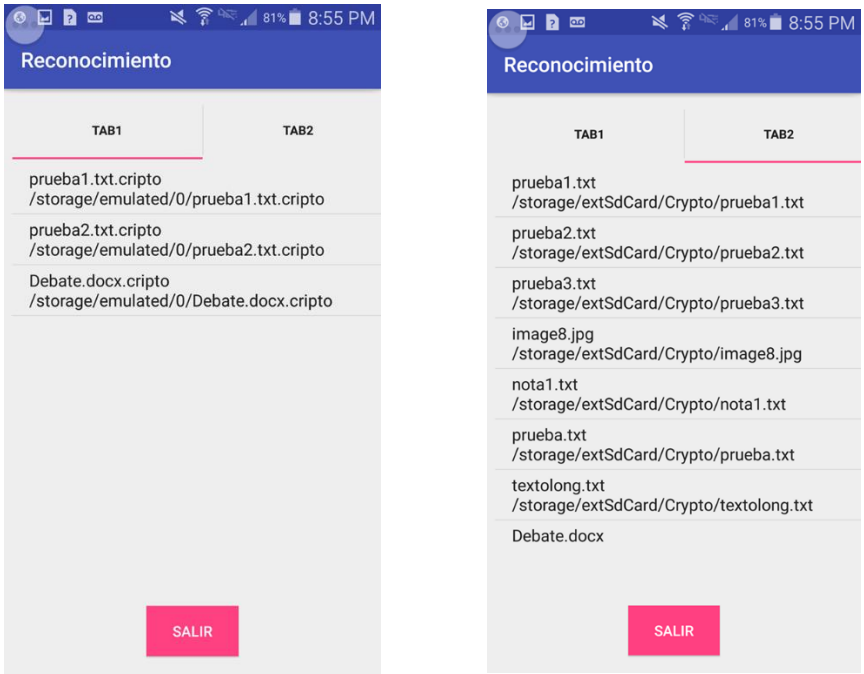

Figura 4.6 Listas de selección de archivos.

Encriptación y Desencriptación.

Para esta sección, el principio de ambas tareas es el mismo. Primeramente se deben leer los archivos a manera de bytes, luego se hace uso de la librería `javax.crypto` la cual contiene las funciones necesarias para llevar a cabo al encriptación y desencriptación de los archivos. Finalmente creamos los archivos en que se almacenaran los bytes transformados.

- Lectura archivo:

Al momento en que el usuario seleccione una casilla dentro de la ListView, se crea una variable string en la que se guarda la ruta en que se encuentra almacenado el archivo. Se hacen uso de los objetos `FileInputStream` y `DataInputStream`, el primero crea un contenedor en el que se guardaran temporalmente los bytes leídos de cada archivo; el segundo nos da acceso a la función `read()` que es la encargada de leer los elementos del archivo.

```
String seleccion1 = mAdapter1.getItem(position);
seleccion1=seleccion1.split("\n")[1];
FileInputStream fis = null;
DataInputStream entrada = null;
byte[] Archivo=null;

try {
    java.io.File fichero = new java.io.File(seleccion1);
    fis = new FileInputStream(fichero);
    entrada = new DataInputStream(fis);
    Archivo = new byte[(int)fichero.length()];
    fis.read(Archivo);
    tam = (int)fichero.length();
}catch (IOException e) {
    Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
}
```

Cuadro 4.13 Lectura de archivos.

- Creación la clave del algoritmo de encriptación y su almacenamiento:

Una vez leídos los bytes del archivo que se desea encriptar o desencriptar, primero se genera llave (*key*) que como se vio en el apartado *3.1.3*, *RC4* utiliza un vector *S* de 256 elementos con todos los números del 0 al 255. Este vector se utilizará como entrada en la siguiente etapa del algoritmo para generar el flujo de cifrado.
Para ello hacemos uso de la `generateKey()` en el que se indica cual algoritmo de encriptación se va a implementar y el tamaño del vector.
Una vez creada la almacenamos en un archivo *.dat* para que posteriormente se puedan desencriptar los archivos una vez se haya cerrado la aplicación.

```
KeyGenerator kg = KeyGenerator.getInstance("RC4");
kg.init(256);
SecretKey sk = kg.generateKey();
ObjectOutputStream escribiendoFichero = new ObjectOutputStream(
        new FileOutputStream("/storage/emulated/0/objetos.dat") );
escribiendoFichero.writeObject(sk);
escribiendoFichero.close();
```

Cuadro 4.14 Creación de llave del algoritmo.

- Función Encriptación/Desencriptación:

Como se puede ver en la siguiente sección de código, la parte de encriptación y desencriptación son similares. Se lee el archivo en el que hemos guardado anteriormente nuestra SecretKey y con el uso de una variable *Cipher* (cifrado) indicamos el método a emplear: ***ENCRYPT_MODE*** o ***DECRYPT_MODE***. El resultado obtenido es un arreglo con los bytes del archivo.

```
//ENCRIPTACION
ObjectInputStream leyendoFichero = new ObjectInputStream(
        new FileInputStream("/storage/emulated/0/objetos.dat") );
SecretKey sk = ( SecretKey ) leyendoFichero.readObject();
leyendoFichero.close();
Cipher cipher = Cipher.getInstance("RC4");
cipher.init(Cipher.ENCRYPT_MODE,sk);
byte[] encriptado = cipher.doFinal(c.getBytes());

//DESENCRIPTACION
ObjectInputStream leyendoFichero = new ObjectInputStream(
        new FileInputStream("/storage/emulated/0/objetos.dat") );
SecretKey sk = ( SecretKey ) leyendoFichero.readObject();
leyendoFichero.close();
Cipher dcipher = Cipher.getInstance("RC4");
dcipher.init(Cipher.DECRYPT_MODE,sk);
byte[]  desencriptado = dcipher.doFinal(Archivo);
```

Cuadro 4.15 Algoritmo RC4.

- Escritura de Archivo.

Por último, una vez obtenido nuestra serie de bytes encriptados o desencriptados, se crea el archivo y se almacena en el teléfono. Para ello usamos las funciones inversas que utilizamos para la Lectura de archivo, que son `FileOutputStream` y `DataOutputStream`.

```
FileOutputStream fos = null;
DataOutputStream salida = null;
String[] parte = seleccion2.split("/");
File path=mem_ext();
String seleccion =path.getAbsolutePath() +"/"+ parte[parte.length - 1]+".cripto";

if(path!=null) {
    fos = new FileOutputStream(seleccion);
    salida = new DataOutputStream(fos);
    salida.write(encriptado);
    fos.close();
    salida.close();
}
```

Cuadro 4.16 Escritura de archivos.

Pantallas de Acceso

En esta última sección se muestran las pantallas que el usuario visualizara entre la transición del menú principal y la lista de archivos. Estas dos interfaces son utilizadas para que el usuario pueda conocer el valor de la frecuencia fundamental de su audio de acceso, además de indicarle si ha sido reconocido por la aplicación o no.
Para estas pantallas se hiso uso de *TimerTask*, que son pantallas emergentes que solo duran unos cuantos segundos en la pantalla.

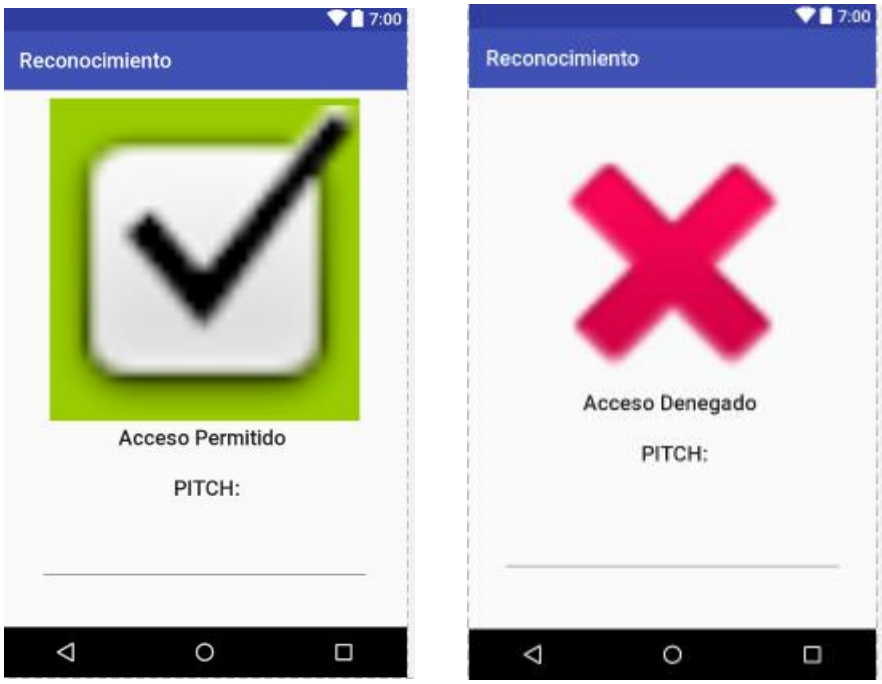

Figura 4.7 Pantallas de acceso.

```
TimerTask task = new TimerTask() {
    @Override
    public void run() {
        Intent intent =
                new Intent(Main5Activity.this, Main3Activity.class);
        startActivity(intent);
    }
};
Timer timer = new Timer();
timer.schedule(task, 3000);
```

Cuadro 4.17 TimerTask.

CAPÍTULO 5. Pruebas y Resultados

En este Capítulo se mostrarán los resultados obtenidos al realizar pruebas con la aplicación Android de reconocimiento de voz, para poder obtener y comparar los objetivos alcanzados, las ventajas y desventajas del proyecto propuesto, así como el porcentaje de éxito.

Para realizar las siguientes pruebas, se usa la voz de cuatro locutores utilizando un teléfono celular marca Samsung® Galaxy S5 SM-G900V dentro de un ambiente no controlado. Estas pruebas consisten en la comparación de los valores obtenidos en el calculo de la frecuencia fundamental y su comparación con las muestras obtenidas en el registro.

Tabla 5.1 Características de los locutores.

Locutor	Genero	Edad	Contraseña
1	Mujer	23 años	moda
2	Mujer	21 años	hola
3	Mujer	22 años	sola
4	Hombre	21 años	soda

Para obtener el rango del Pitch de cada locutor, se realizaron 50 pruebas utilizando diferentes contraseñas propuestas, colocando el celular a una distancia de aproximadamente 20cm de la boca, que es lo recomendado por la misma aplicación. Una vez definido el rango del pitch, se determinó la probabilidad de éxito o fracaso del sistema, para ello se requirió hacer un análisis estadístico con los resultados obtenidos por cada uno de los locutores.
Al obtener las tres muestras del registro se calcula el promedio. Para que el locutor sea capaz de acceder a la aplicación, el pitch de la grabación de acceso debe estar dentro de un rango entre -15 a +15 Hertz del valor promedio. En las siguientes tablas se muestra el valor del pitch de las grabaciones para acceder a la aplicación una vez teniendo su registro, las casillas marcadas en rojo son las pruebas en que el locutor no pudo acceder.

Figura 5.1 Celular empleado para las pruebas de la aplicación.

A continuación, se muestran las pruebas realizadas con nuestros cinco locutores. Cada locutor tendrá su propio registro, la primera tabla muestra el intento por entrar a la aplicación, mientras que las otras cuatro serán los intentos de los locutores restantes al tratar de acceder a la aplicación con un registro diferente al suyo.

5.1 Pruebas registro Locutor 1.

Tabla 5.2 Pitch de los audios muestra del registro de Locutor 1.

Muestra	Pitch
1	329
2	289
3	324

Pitch promedio las muestras del Locutor 1: 314 Hz.

Tabla 5.3 Valor del pitch del audio de acceso de Locutor 1.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	316	11	311	21	324	31	320	41	309
2	311	12	319	22	305	32	316	42	300
3	345	13	303	23	309	33	329	43	314
4	309	14	279	24	386	34	316	44	306
5	295	15	314	25	305	35	300	45	316
6	316	16	307	26	305	36	309	46	309
7	300	17	242	27	311	37	314	47	299
8	324	18	314	28	368	38	316	48	303
9	319	19	305	29	324	39	320	49	320
10	299	20	320	30	309	40	316	50	314

Tabla 5.4 Intentos de acceso del Locutor 2.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	287	11	279	21	291	31	279	41	272
2	294	12	272	22	305	32	287	42	286
3	260	13	287	23	286	33	287	43	284
4	290	14	292	24	295	34	290	44	280
5	295	15	266	25	300	35	300	45	287
6	270	16	279	26	276	36	296	46	293
7	285	17	269	27	279	37	273	47	280
8	285	18	283	28	262	38	284	48	274
9	302	19	287	29	285	39	287	49	285
10	290	20	311	30	293	40	266	50	292

Tabla 5.5 Valor del pitch del audio de acceso de Locutor 3.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	243	11	291	21	256	31	265	41	264
2	268	12	279	22	264	32	261	42	258
3	252	13	258	23	271	33	259	43	267
4	265	14	245	24	264	34	263	44	265
5	271	15	287	25	265	35	244	45	267
6	280	16	299	26	270	36	259	46	271
7	278	17	253	27	255	37	264	47	264
8	264	18	266	28	269	38	267	48	269
9	277	19	259	29	281	39	267	49	267
10	292	20	260	30	272	40	270	50	266

Tabla 5.6 Valor del pitch del audio de acceso de Locutor 4.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	237	11	236	21	233	31	242	41	229
2	248	12	248	22	237	32	237	42	233
3	239	13	253	23	241	33	233	43	239
4	224	14	245	24	266	34	221	44	229
5	233	15	247	25	245	35	237	45	234
6	241	16	231	26	239	36	239	46	233
7	239	17	238	27	234	37	240	47	228
8	243	18	232	28	237	38	224	48	236
9	248	19	229	29	227	39	220	49	234
10	230	20	234	30	229	40	233	50	241

Tabla 5.7 Valores de tendencia central del Locutor 1.

Media	312.8
Mediana	312.5
Desviación	19.37
Varianza	375.51
Moda	316
Porcentaje de éxito	91%

5.2 Pruebas registro Locutor 2.

Tabla 5.8 Pitch de los audios muestra del registro de Locutor 2.

Muestra	Pitch
1	295
2	289
3	277

Pitch promedio del Locutor 2: 287 Hz.

Tabla 5.9 Valor del pitch del audio de acceso de Locutor 2.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	292	11	293	21	291	31	279	41	290
2	266	12	280	22	305	32	287	42	303
3	279	13	274	23	286	33	287	43	287
4	290	14	287	24	295	34	290	44	280
5	295	15	280	25	279	35	287	45	287
6	270	16	287	26	271	36	287	46	293
7	280	17	290	27	287	37	270	47	280
8	287	18	305	28	292	38	285	48	274
9	293	19	287	29	285	39	287	49	285
10	280	20	311	30	293	40	311	50	292

Tabla 5.10 Intentos de acceso del Locutor 1.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	316	11	321	21	329	31	320	41	279
2	311	12	319	22	305	32	316	42	314
3	306	13	303	23	309	33	329	43	314
4	316	14	309	24	386	34	307	44	306
5	309	15	300	25	309	35	242	45	316
6	316	16	314	26	316	36	314	46	309
7	300	17	306	27	305	37	305	47	299
8	324	18	309	28	309	38	316	48	303
9	319	19	305	29	324	39	320	49	279
10	299	20	320	30	309	40	316	50	314

Tabla 5.11 Intentos de acceso del Locutor 3.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	265	11	291	21	269	31	265	41	244
2	267	12	279	22	258	32	261	42	259
3	271	13	258	23	267	33	259	43	267
4	264	14	256	24	264	34	263	44	265
5	271	15	264	25	277	35	265	45	267
6	280	16	271	26	270	36	291	46	271
7	278	17	264	27	255	37	279	47	264
8	264	18	265	28	269	38	258	48	269
9	270	19	259	29	281	39	267	49	267
10	292	20	260	30	272	40	270	50	266

Tabla 5.12 Intentos de acceso del Locutor 4.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	245	11	229	21	233	31	242	41	229
2	239	12	233	22	236	32	234	42	233
3	234	13	253	23	234	33	233	43	239
4	224	14	245	24	234	34	228	44	229
5	233	15	234	25	245	35	245	45	234
6	241	16	233	26	239	36	234	46	233
7	239	17	228	27	234	37	233	47	228
8	229	18	236	28	269	38	253	48	236
9	233	19	234	29	245	39	245	49	234
10	239	20	234	30	239	40	234	50	241

Tabla 5.13 Valores de tendencia central del Locutor 2.

Media	287.02
Mediana	287
Desviación	9.65
Varianza	93.2
Moda	287
Porcentaje de éxito	93%

5.3 Pruebas registro Locutor 3.

Tabla 5.14 Pitch de los audios muestra del registro de Locutor 3.

Muestra	Pitch
1	237
2	288
3	269

Pitch promedio del Locutor 3: 264 Hz.

Tabla 5.15 Valor del pitch del audio de acceso de Locutor 3.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	264	11	291	21	269	31	271	41	265
2	265	12	269	22	258	32	264	42	267
3	291	13	265	23	267	33	269	43	255
4	279	14	267	24	264	34	263	44	269
5	271	15	271	25	277	35	265	45	281
6	280	16	264	26	270	36	248	46	267
7	278	17	263	27	250	37	279	47	270
8	264	18	265	28	269	38	258	48	269
9	270	19	287	29	281	39	267	49	267
10	292	20	260	30	272	40	270	50	266

Tabla 5.16 Intentos de acceso del Locutor 1.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	305	11	321	21	329	31	316	41	314
2	316	12	319	22	305	32	320	42	242
3	320	13	316	23	309	33	316	43	314
4	316	14	320	24	299	34	314	44	305
5	309	15	316	25	303	35	242	45	316
6	299	16	314	26	279	36	314	46	309
7	300	17	309	27	314	37	300	47	299
8	324	18	324	28	309	38	324	48	303
9	319	19	309	29	324	39	319	49	279
10	299	20	320	30	329	40	299	50	314

Tabla 5.17 Intentos de acceso del Locutor 2.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	292	11	287	21	291	31	279	41	290
2	266	12	280	22	305	32	287	42	303
3	279	13	287	23	286	33	287	43	287
4	290	14	293	24	290	34	280	44	280
5	295	15	280	25	305	35	291	45	287
6	270	16	287	26	271	36	305	46	293
7	280	17	290	27	287	37	286	47	280
8	287	18	305	28	292	38	287	48	274
9	293	19	287	29	285	39	287	49	285
10	288	20	311	30	293	40	311	50	292

Tabla 5.18 Intentos de acceso del Locutor 4.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	234	11	245	21	237	31	242	41	229
2	233	12	234	22	245	32	234	42	233
3	228	13	233	23	234	33	233	43	239
4	233	14	245	24	234	34	228	44	229
5	228	15	234	25	245	35	245	45	234
6	236	16	233	26	239	36	234	46	233
7	234	17	228	27	234	37	233	47	228
8	229	18	236	28	237	38	253	48	236
9	233	19	234	29	245	39	245	49	234
10	239	20	234	30	239	40	234	50	241

Tabla 5.19 Valores de tendencia central del Locutor 3.

Media	269.26
Mediana	268
Desviación	9.28
Varianza	86.27
Moda	269
Porcentaje de éxito	93%

5.4 Pruebas registro Locutor 4.

Tabla 5.20 Pitch de los audios muestra del registro de Locutor 4.

Muestra	Pitch
1	244
2	232
3	224

Pitch promedio del Locutor 4: 233 Hz.

Tabla 5.21 Valor del pitch del audio de acceso de Locutor 4.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	228	11	245	21	237	31	242	41	259
2	255	12	234	22	255	32	234	42	233
3	234	13	233	23	234	33	215	43	233
4	233	14	253	24	234	34	228	44	228
5	228	15	245	25	236	35	245	45	236
6	236	16	233	26	234	36	253	46	233
7	234	17	228	27	241	37	245	47	228
8	218	18	236	28	228	38	233	48	236
9	233	19	215	29	236	39	228	49	217
10	239	20	234	30	239	40	234	50	245

Tabla 5.22 Intentos de acceso del Locutor 1.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	314	11	324	21	330	31	279	41	299
2	305	12	303	22	305	32	319	42	303
3	279	13	279	23	309	33	316	43	314
4	319	14	314	24	299	34	316	44	319
5	319	15	305	25	303	35	320	45	299
6	299	16	316	26	316	36	316	46	300
7	300	17	320	27	309	37	279	47	324
8	324	18	316	28	300	38	316	48	303
9	319	19	300	29	324	39	320	49	279
10	302	20	324	30	329	40	316	50	319

Tabla 5.23 Intentos de acceso del Locutor 2.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	290	11	287	21	291	31	279	41	290
2	305	12	280	22	305	32	287	42	303
3	271	13	287	23	280	33	292	43	287
4	287	14	287	24	287	34	280	44	280
5	292	15	280	25	281	35	287	45	287
6	285	16	287	26	280	36	287	46	293
7	280	17	292	27	274	37	293	47	280
8	287	18	266	28	285	38	280	48	274
9	293	19	279	29	287	39	274	49	285
10	288	20	290	30	293	40	285	50	292

Tabla 5.24 Intentos de acceso del Locutor 3.

Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch	Prueba	Pitch
1	250	11	291	21	258	31	269	41	265
2	269	12	249	22	267	32	281	42	267
3	281	13	265	23	267	33	269	43	255
4	272	14	267	24	264	34	263	44	269
5	271	15	271	25	277	35	265	45	264
6	280	16	269	26	270	36	248	46	277
7	278	17	255	27	250	37	279	47	270
8	264	18	269	28	269	38	277	48	269
9	270	19	264	29	281	39	291	49	267
10	292	20	279	30	272	40	279	50	266

Tabla 5.25 Valores de tendencia central del Locutor 4.

Media	235.44
Mediana	234
Desviación	8.68
Varianza	75.35
Moda	234
Porcentaje de éxito	91%

5.5 Estudio Económico

La realización de un estudio económico adecuado sobre el tamaño y esfuerzo requerido es una de las características fundamentales de un proyecto exitoso.

El estudio está asociado con el costo y el tiempo de las actividades identificadas en el proyecto. El objetivo del estudio del proyecto es reducir los costos e incrementar los niveles de servicio y calidad.

Tabla 5.26 Costos del Proyecto

Cantidad	Concepto	Costo (MXN)	Frecuencia de pago
1	Android Studio	0.00	-
1	Windows 10 Home	2,888.00	Única
1	Computadora	16,400	Única
-	Servicio de internet	400	Mensual
-	Ingeniero (Desarrollador)	9000	Mensual

Para obtener el salario mínimo mensual que percibe un desarrollador de software en México se toma como referencia la información brindada por el portal de empleos OCC México.

Una vez teniendo estos conceptos, debemos considerar el tiempo en que se desarrollo el proyecto para sacar el costo total de los pagos que se realizan mensualmente. Tomando en cuenta que el proyecto se llevo a cabo en el lapso de 1 año, los costos totales correspondientes al pago del servicio de internet y el sueldo del desarrollador del software son multiplicados por los 12 meses dándonos un total de $4,800.00 y $108,000.00 respectivamente.
Finalmente tenemos que el costo total del proyecto es de **$132,088.00 MXN.**

Capítulo 6. Conclusiones y Trabajo Futuro

6.1 Conclusiones

La implementación de un sistema de reconocimiento de voz permitió acceder de manera más rápida y sencilla a la aplicación, siendo que el uso de autocorrelación en nuestro sistema no mostró ser 100% efectiva, pero se encontró dentro de los estándares para ser considerada. Dentro de las pruebas todos lo interlocutores fueron reconocidos correctamente el 90% de las veces con su propio registro; en el caso de los intentos de locutores externos algunas muestras fueron capaces accesar a la aplicación.

El acceso de locutores a un registro que no era propio de ellos se debió a varios factores, como el ruido del ambiente, las distancia a la que el teléfono celular se encontraba del usuario y el volumen con el que se mencionaba la contraseña.

Otro punto importante fue la selección de contraseñas para el acceso, el uso de frases completas o palabras largas hace más complicado el reconocimiento de voz. es por eso que en las pruebas fueron escogidas palabras de menor tamaño, lo que permitió aumentar la probabilidad de éxito del reconocimiento de voz.

Observando los resultados cabe mencionar que la función de autocorrelación fue adecuada con respecto al tiempo de procesamiento del teléfono celular, además de que su implementación logró ser sencilla dentro del lenguaje de programación que se utilizó.

Por otra parte, el algoritmo que se implementó en la aplicación resulto ser exitoso en los archivos de texto; imposibilito a los usuarios externos poder ver el contenido de los archivos encriptados por la aplicación. El problema que se encontró fue al momento en desencriptar otro tipo de archivos como imágenes y videos, el sistema fue capaz de encriptarlos, pero no de recuperar su formato original.

El tamaño de los archivos generados al momento de encriptar fue el mismo que el del archivo original por lo que no se consumió una mayor cantidad de memoria y no hubo perdida de información en la transformación.

El uso del algoritmo RC4 no afecto en gran cantidad a la aplicación, el tiempo de procesamiento varió dependiendo del tamaño del archivo original, es por eso que la aplicación se limitó solo a archivos de texto.

El software de Android Studio brindo la oportunidad de diseñar nuestro sistema de una manera más práctica y que el tiempo de desarrollo pudiera optimizarse.

La interfaz para el manejo de documentos permitió a los usuarios visualizar los archivos capaces de ser encriptados o desencriptados. Los usuarios lograron hacer uso de la encriptación de archivos con solo un click en la interfaz. Sin embargo, el mismo sistema Android no permitió que algunas de las tareas de la aplicación se pudieran ejecutar dentro de la memoria externa del teléfono celular, por lo que se tuvo que también se empleó la memoria interna.

La aplicación no funcionó correctamente en dispositivos diferentes al celular utilizado en las pruebas. Esto se debió a que las carpetas en las que se almacenan los audios de registro son nombradas de diferente manera, dependiendo de la versión Android. Además que el calculo de la frecuencia fundamental se basa en la frecuencia de muestro a la que graba el micrófono integrado al teléfono celular, lo que también cambia dependiendo el modelo y marca del dispositivo.

6.2 Trabajo Futuro

Tratándose de un proyecto enfocado a la tecnología, la cual cambia contantemente, se pueden realizar mejoras para aumentar el porcentaje de fiabilidad del reconocimiento de voz o poder optimizar el procesamiento en las tareas que lleva a cabo la aplicación.

Algunos de los proyectos a futuro se encuentran:

- Realizar una base de datos en internet para el respaldo de los archivos encriptados.
- Aplicar otra función matemática del procesamiento de señales, por ejemplo, coeficientes de predicción lineal, para que el reconocimiento de voz sea más confiable y que además permita el uso de contraseñas de mayor longitud.
- Implementar un dispositivo de grabación que se adapte al celular para que los audios obtenidos no contengan ruido.
- Utilizar diferentes algoritmos de encriptación, como DES o AES que brinde un mayor nivel de seguridad al momento de encriptar los archivos.
- Lograr que la aplicación maneje otro tipo de archivos, como audio o video.
- Adaptar el código a diferentes marcas y modelos de celulares.
- Que la aplicación sea capaz de calcular otras características de la voz del usuario.

Referencias

[1] Ortega Triguero, Jesús J. “Introducción a la Criptografía: Historia y Actualidad”. Editorial Monografías 2006.
[2] Baca Urbania, Gabriel. “Introducción a la Seguridad Informática”. Editorial Patria 2016.
[3] Carballar Falcón, José Antonio. “WI-FI: Lo que se necesita conocer”. Editorial RC LIBROS 2010.
[4] Pérez Ramírez, Fredy O. “Introducción a las Series de Tiempo”. Editorial Universidad de Medellin 2007.
[5] Rabiner, Johanson. “Digital Processing Of Speech Singanls” Editorial Prince- Hall 2010.
[6] Rabiner, Lawrence R. “On the Use of Autocorrelation Analysis for Pitch Detector” IEEE 1977.
[7] Amaro Soriano, José Enrique. “Android. Programación de dispositivos móviles a través de ejemplos”. Editorial Marcombo 2012
[8]https://developer.android.com/reference
[9]https://www.researchgate.net/publication/306024204_A_tutorial_to_extract_the_pitch_in_speech_signals_using_autocorrelation
[10]http://www.tutorialesprogramacionya.com/javaya/androidya
[11] http://chuwiki.chuidiang.org/index.php?title=Lectura_y_Escritura_de_Ficheros_en_Java
[12] https://stackoverflow.com/questions/12289717/rc4-encryption-java

ANEXOS

Anexo 1. Menú principal

```
package com.example.hp.reconocimiento;

import android.content.Intent;
import android.net.Uri;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;

public class MainActivity extends AppCompatActivity {
    ImageButton acceso;
    Button registro;
    int peticion=1;
    Uri url1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //GENERACION DE CLAVE
        try {
            KeyGenerator kg = KeyGenerator.getInstance("RC4");
            kg.init(256);
            SecretKey sk = kg.generateKey();
            ObjectOutputStream escribiendoFichero = new ObjectOutputStream(
                    new FileOutputStream("/storage/emulated/0/objetos.dat") );
            escribiendoFichero.writeObject(sk);
            escribiendoFichero.close();
        }catch (Exception e){
            Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
        }

        //CREACION DE BOTONES
        acceso = (ImageButton) findViewById(R.id.imageButton);
        registro = (Button) findViewById(R.id.button);

        //REGISTRO
        registro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent =
                        new Intent(MainActivity.this, Main2Activity.class);
                //INICIAMOS LA NUEVA ACTIVIDAD
```

```
                startActivity(intent);
            }
        });

        //ACCESO
        acceso.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
                startActivityForResult(intent, peticion);
            }
        });
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK && requestCode == peticion) {
            url1 = data.getData();
        }
        //Lectura de Audios y Calculo de Frec Fundamental
        double F1 = leerFichero("/storage/emulated/0/Sounds/Nota 001.m4a");
        double F2 = leerFichero("/storage/emulated/0/Sounds/Nota 002.m4a");
        double F3 = leerFichero("/storage/emulated/0/Sounds/Nota 003.m4a");
        double FU = leerFichero("/storage/emulated/0/Sounds/Nota 004.m4a");
        double Prom = (F1+F2+F3)/3;
        if (FU>Prom)
            FU = FU-10;
        else
            FU = FU+10;
        double RecUs = Math.abs(FU-Prom);

        //Almacenamiento valor PITCH
        try{
            FileOutputStream fos = null;
            DataOutputStream salida = null;
            fos = new FileOutputStream("/storage/emulated/0/Sounds/pt.txt");
            salida = new DataOutputStream(fos);
            salida.writeDouble(FU);
            fos.close();
            salida.close();
        }catch(Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(),
           Toast.LENGTH_SHORT).show();
        }


        //Reconocimiento
        if (RecUs<15){
            Intent intent2 =
                    new Intent(MainActivity.this, Main5Activity.class);
            startActivity(intent2);
        }else{
            Toast.makeText(this,""+F1+", "+F2+", "+F3+",
"+Prom,Toast.LENGTH_LONG).show();
            Intent intent =
                    new Intent(MainActivity.this, Main4Activity.class);
            startActivity(intent);
        }
    }

    private double leerFichero(String Ruta){
        FileInputStream fis=null;
        DataInputStream entrada=null;
```

```
        int tam=0,p=0;
        double mayor=0;

        try {
            //Verificacion del tamaño de la señal
            fis = new FileInputStream(Ruta);
            entrada = new DataInputStream(fis);
            while(entrada.readDouble() != 0) {
                tam++;
            }
            entrada.close();
            fis.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
        //LECTURA DEL AUDIO
        double[] datos=new double[tam];
        int i=0;
        try {
            fis = new FileInputStream(Ruta);
            entrada = new DataInputStream(fis);
            while(i<tam){
                datos[i]=entrada.readDouble();
                i++;
            }
            entrada.close();
            fis.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }

        //AUTOCORRELACION
        double[] acf = new double[tam];
        double suma = 0,s=0;
        int Tam = 1000;
            for (int k = 0; k < Tam; k++) {
            suma = 0;
            for (int n = 0; n < Tam - k; n++) {
                s = (datos[n] * datos[n + k])*Math.pow(10,100);
                if (s<Math.pow(10,250) && s>Math.pow(10,-250)){
                    suma = suma + s;
                }
            }
            acf[k] = (suma / Tam)/Math.pow(10,244);
        }

        //CALCULO DE FREC FUNDAMENTAL Y EXTRACCION DEL PITCH
        int Rmin = 44100/350; //126
        int Rmax = 44100/50; //882
        for (int j=Rmin; j<Rmax ; j++)
        {
            if (acf[j] >= mayor) {
                mayor = acf[j];
                p=j;
            }
        }
        float ff = 44100/p;
        return ff;
    }
}
```

Código XML del menú principal

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.hp.reconocimiento.MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="213dp"
        android:layout_height="90dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:text="          ¡BIENVENIDO!                    En caso de ser su
primera vez haga click en el boton Registrar"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.091" />

    <Button
        android:id="@+id/button"
        android:layout_width="123dp"
        android:layout_height="61dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:background="?android:attr/colorActivatedHighlight"
        android:text="REGISTRAR"

android:textAppearance="@style/TextAppearance.AppCompat.Light.SearchResult.Title"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.322"
        app:layout_constraintHorizontal_bias="0.502" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="208dp"
        android:layout_height="73dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:text="Si ya cuenta con su contraseña, presione el microfono para
ingresarla"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.634"
        app:layout_constraintHorizontal_bias="0.503"
        tools:layout_editor_absoluteY="276dp"
        tools:layout_editor_absoluteX="93dp" />

    <ImageButton
        android:id="@+id/imageButton"
        android:layout_width="72dp"
        android:layout_height="76dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:drawable/presence_audio_online"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.837"
        app:srcCompat="@android:color/transparent" />
</android.support.constraint.ConstraintLayout>
```

Anexo 2. Menú registro

```
package com.example.hp.reconocimiento;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.content.Intent;
import android.provider.MediaStore;
import android.net.Uri;

public class Main2Activity extends AppCompatActivity {
    ImageButton rec1;
    ImageButton rec2;
    ImageButton rec3;
    Button siguiente;
    int peticion=1;
    Uri url1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        //CREACION DE BOTONES
        rec1 = (ImageButton) findViewById(R.id.imageButton2);
        rec2 = (ImageButton) findViewById(R.id.imageButton3);
        rec3 = (ImageButton) findViewById(R.id.imageButton4);
        siguiente = (Button) findViewById(R.id.button2);

        //GRABACION DE MUESTRAS
        rec1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
```

```
                Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
                startActivityForResult(intent, peticion);
            }
        });
        rec2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
                startActivityForResult(intent, peticion);
            }
        });
        rec3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
                startActivityForResult(intent, peticion);
            }
        });

        //SIGUIENTE
        siguiente.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent =
                        new Intent(Main2Activity.this, MainActivity.class);
                //INICIAMOS LA NUEVA ACTIVIDAD
                startActivity(intent);
            }
        });
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK && requestCode == peticion) {
            url1 = data.getData();
        }
    }
}
```

Código XML del menú registro

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.hp.reconocimiento.Main2Activity">

    <ImageButton
        android:id="@+id/imageButton2"
        android:layout_width="58dp"
        android:layout_height="62dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:drawable/presence_audio_online"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.528"
        app:srcCompat="@android:color/transparent"
        app:layout_constraintHorizontal_bias="0.145" />

    <ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="58dp"
        android:layout_height="62dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:drawable/presence_audio_online"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.528"
        app:srcCompat="@android:color/transparent" />

    <ImageButton
        android:id="@+id/imageButton4"
        android:layout_width="58dp"
        android:layout_height="62dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:drawable/presence_audio_online"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.832"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.528"
        app:srcCompat="@android:color/transparent" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="225dp"
        android:layout_height="109dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:text="Presione los botones para crear su contraseña
(Debe mencionar su frase clave de la manera mas clara posible)"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.148" />

    <Button
        android:id="@+id/button2"
        android:layout_width="127dp"
        android:layout_height="68dp"
```

```
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:backgroundTint="@android:color/holo_orange_dark"
        android:text="SIGUIENTE"

android:textAppearance="@style/TextAppearance.AppCompat.Light.SearchResult.Title"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.897"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.873"
        tools:layout_editor_absoluteX="223dp" />

</android.support.constraint.ConstraintLayout>
```

Anexo 3. Listas de selección y encriptación de archivos

```
package com.example.hp.reconocimiento;

import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.content.Intent;
import android.widget.Button;
import android.widget.TabHost;
import android.widget.ListView;
import android.widget.Toast;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectInputStream;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;

public class Main3Activity extends AppCompatActivity {
    TabHost tabs;
    int peticion=1, tam=0;
    Button salir;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);

        salir = (Button)findViewById(R.id.button4);
        //Creacion de Tabs
```

```
        tabs = (TabHost)findViewById(R.id.tabhost); //vinculacion de la parte visual
con el tabs
        tabs.setup();

        //Salida de la aplicacion
        salir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent =
                        new Intent(Main3Activity.this, MainActivity.class);
                startActivity(intent);
            }
        });

        //Creacion de listas
        final ArrayAdapter<String> mAdapter1;
        mAdapter1=new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1);
        final ArrayAdapter<String> mAdapter2;
        mAdapter2=new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1);

        //Busqueda de archivos para encriptar
        File miDir=new File ("/storage/extSdCard/Crypto");
        try {
            File[] files = miDir.listFiles();
            for (File file : files) {
                if (!file.isDirectory()) {
                    if(file.getName().contains(".cripto")){
                    }else{
                        mAdapter2.add(file.getName()+"\n"+file.getAbsolutePath());
                    }
                }
            } }
        catch (Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
        }
        //Busqueda de archivos para desencriptar
        File miDirCrip=new File ("/storage/emulated/0");
        try {
            File[] files = miDirCrip.listFiles();
            for (File file : files) {
                if (!file.isDirectory()) {
                    if(file.getName().contains(".cripto")){
                        mAdapter1.add(file.getName()+"\n"+file.getAbsolutePath());
                    }
                }
            } }
        catch (Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
        }
        //Impresion de Listas
        final ListView listView=(ListView) findViewById(R.id.listView);
        listView.setAdapter(mAdapter1);
        final ListView listView2=(ListView) findViewById(R.id.listView2);
        listView2.setAdapter(mAdapter2);

        TabHost.TabSpec spec1 = tabs.newTabSpec("Tab1");
        spec1.setIndicator("Tab1");
        spec1.setContent(R.id.listView);
        tabs.addTab(spec1);
```

```
        TabHost.TabSpec spec2 = tabs.newTabSpec("Tab2");
        spec2.setIndicator("Tab2");
        spec2.setContent(R.id.listView2);
        tabs.addTab(spec2);
         //escoger archivos para guardar en la app
       listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                                        @Override
                                        public void onItemClick (AdapterView < ? >
       adapter, View view,int position, long arg) {
// TODO Auto-generated method stub
String seleccion1 = mAdapter1.getItem(position);
seleccion1=seleccion1.split("\n")[1];
FileInputStream fis = null;
DataInputStream entrada = null;
byte[] Archivo=null;

//LECTURA DE ARCHIVO
try {
    java.io.File fichero = new java.io.File(seleccion1);
    fis = new FileInputStream(fichero);
    entrada = new DataInputStream(fis);
    Archivo = new byte[(int)fichero.length()];
    fis.read(Archivo);
    tam = (int)fichero.length();
}catch (EOFException e) {
    //Toast.makeText(getApplicationContext(), "byte"+archivo,
Toast.LENGTH_LONG).show();
}catch (FileNotFoundException e) {
    Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
}catch (IOException e) {
    Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
}
try {
    entrada.close();
    fis.close();
}catch(IOException e) {
}

//DESENCRIPTACION
try {
    ObjectInputStream leyendoFichero = new ObjectInputStream(
            new FileInputStream("/storage/emulated/0/objetos.dat") );
    SecretKey sk = ( SecretKey ) leyendoFichero.readObject();
    leyendoFichero.close();

    Cipher dcipher = Cipher.getInstance("RC4");
    dcipher.init(Cipher.DECRYPT_MODE,sk);
    byte[]  desencriptado = dcipher.doFinal(Archivo);
    String p="",r="";
    char c = 0;
    int b = 0;
    //char caract=0;
    for (int j=0;j<tam;j++)
    {
        p = Byte.toString(desencriptado[j]);
        b = Integer.parseInt(p);
        c = (char)b;
        r = r+c;
    }

    //CREACION DE ARCHIVO ORIGINAL
    String[] parte = seleccion1.split("/");
    String[] name = parte[4].split(".cripto");
```

```
    File path=mem_ext();
    String seleccion =path.getAbsolutePath() +"/" +name[0];
    FileWriter fw;
    fw = new FileWriter(seleccion);
    fw.write(r);
    fw.close();
}
catch(FileNotFoundException e){
    Toast.makeText(getApplicationContext(), e.getMessage(),Toast.LENGTH_LONG).show();
} catch(IOException e){
    Toast.makeText(getApplicationContext(), e.getMessage(),Toast.LENGTH_LONG).show();
}catch(Exception e){
    Toast.makeText(getApplicationContext(), e.getMessage(),Toast.LENGTH_LONG).show();
}
//Recargar Activity
Intent intent =
        new Intent(Main3Activity.this, Main3Activity.class);
startActivity(intent);
Toast.makeText(getApplicationContext(), "Archivo Desencriptado",
Toast.LENGTH_SHORT).show();
}
};

listView2.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                                    @Override
                                    public void onItemClick (AdapterView < ? >
adapter, View view,int position, long arg){
// TODO Auto-generated method stub
String seleccion2 = mAdapter2.getItem(position);
seleccion2=seleccion2.split("\n")[1];

try {
    //LECTURA DE ARCHIVO
    FileReader fr;
    int caract;
    fr = new FileReader(seleccion2);
    caract = fr.read();
    char a,b;
    a = (char)caract;
    String c = a+"";
    while (caract != -1) {
        caract = fr.read();
        if (caract != -1) {
            b = (char) caract;
            c = c + b;
        }
    }
    //ENCRIPTACION
    ObjectInputStream leyendoFichero = new ObjectInputStream(
            new FileInputStream("/storage/emulated/0/objetos.dat") );
    SecretKey sk = ( SecretKey ) leyendoFichero.readObject();
    leyendoFichero.close();
    Cipher cipher = Cipher.getInstance("RC4");
    cipher.init(Cipher.ENCRYPT_MODE,sk);
    byte[] encriptado = cipher.doFinal(c.getBytes());

    //CREACION DE ARCHIVO ENCRIPTADO
    FileOutputStream fos = null;
    DataOutputStream salida = null;
    String[] parte = seleccion2.split("/");
    File path=mem_ext();
    String seleccion =path.getAbsolutePath() +"/"+ parte[parte.length - 1]+".cripto";
```

```
    if(path!=null) {
        fos = new FileOutputStream(seleccion);
        salida = new DataOutputStream(fos);
        salida.write(encriptado);
        fos.close();
        salida.close();
    }
}
catch(Exception e) {
    Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
}
//Recargar Activity
Intent intent =
        new Intent(Main3Activity.this, Main3Activity.class);
startActivity(intent);
Toast.makeText(getApplicationContext(), "Archivo Encriptado",
Toast.LENGTH_SHORT).show();
}
};

public File mem_ext() {
        boolean sdDisponible = false;
        boolean sdAccesoEscritura = false;
        File path=null;
        //Comprobamos el estado de la memoria externa (tarjeta SD)
        try {
            path = Environment.getExternalStorageDirectory();
            String estado = Environment.getExternalStorageState(path);
            if (estado.equals(Environment.MEDIA_MOUNTED)) {
                sdDisponible = true;
                sdAccesoEscritura = true;
                if (estado.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
                    sdDisponible = true;
                    sdAccesoEscritura = false;
                } else {
                    sdDisponible = false;
                    sdAccesoEscritura = false;
                }
            }
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
        }
        return path;
    }
}
```

Código XML de las listas de selección y encriptación

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.hp.reconocimiento.Main3Activity">

    <TabHost
        android:id="@+id/tabhost"
```

```
    android:layout_width="364dp"
    android:layout_height="430dp"
    android:layout_marginBottom="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="431dp"
        android:orientation="vertical">

        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="match_parent"
            android:layout_height="63dp"
            android:visibility="visible"
            tools:visibility="visible">

        </TabWidget>

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="match_parent"
            android:layout_height="370dp">

            <LinearLayout
                android:id="@+id/encriptados"
                android:layout_width="match_parent"
                android:layout_height="364dp"
                android:orientation="vertical">

                <ListView
                    android:id="@+id/listView"
                    android:layout_width="match_parent"
                    android:layout_height="362dp" />
            </LinearLayout>

            <LinearLayout
                android:id="@+id/desencriptados"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">

                <ListView
                    android:id="@+id/listView2"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent" />
            </LinearLayout>

        </FrameLayout>
    </LinearLayout>
</TabHost>

<Button
    android:id="@+id/button4"
    style="@style/Widget.AppCompat.Button.Colored"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:background="?android:attr/colorAccent"
        android:text="Salir"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.982" />
</android.support.constraint.ConstraintLayout>
```

Anexo 4. Pantallas de acceso

```
package com.example.hp.reconocimiento;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.Toast;

import java.io.DataInput;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.util.Timer;
import java.util.TimerTask;

public class Main4Activity extends AppCompatActivity {
    private EditText pitch;
    double FU;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);

        pitch = (EditText)findViewById(R.id.editText);
        try{
            FileInputStream fis = null;
            DataInput entrada = null;
            fis = new FileInputStream("/storage/emulated/0/Sounds/pt.txt");
            entrada = new DataInputStream(fis);
            FU = entrada.readDouble();
            fis.close();
        }catch(Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
        pitch.setText(Double.toString(FU)+" Hz");
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                Intent intent =
                new Intent(Main4Activity.this, MainActivity.class);
                startActivity(intent);
            }
```

```
        };

        Timer timer = new Timer();
        timer.schedule(task, 3000);
    }
}
```

Código XML de las pantallas de acceso

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.hp.reconocimiento.Main4Activity">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="174dp"
        android:layout_height="76dp"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="8dp"
        android:text="Acceso Denegado \n\n PITCH:"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.SearchResult.Title"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.662" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="319dp"
        android:layout_height="314dp"
        app:srcCompat="@android:drawable/ic_delete"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="8dp"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintVertical_bias="0.0"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintHorizontal_bias="0.509"
        tools:layout_editor_absoluteX="30dp" />

    <EditText
        android:id="@+id/editText"
        style="@android:style/Widget.DeviceDefault.Light.AutoCompleteTextView"
        android:layout_width="322dp"
        android:layout_height="87dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
```

```
        android:ems="10"
        android:inputType="textPersonName"
        android:textSize="50dp"
        android:textAlignment="center"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.894"
        tools:layout_editor_absoluteY="372dp"
        tools:layout_editor_absoluteX="-66dp" />
</android.support.constraint.ConstraintLayout>
```