Apéndice A
Criptografía y Seguridad Informática

La *criptología* (del griego *krypto*: oculto, lo escondido y *logos:* estudio) es la disciplina que trata los problemas relacionados con la seguridad en el intercambio de mensajes en clave entre un emisor y un receptor a través de un canal de comunicaciones. Es una de las ramas más antiguas de la matemática. Hace más de dos mil años, el imperio romano enviaba mensajes secretos durante las campañas militares de forma que si el mensajero era interceptado la información que portaba no corriera el peligro de caer en manos del enemigo. La criptología comprende dos grandes áreas: la *criptografía* que se ocupa del cifrado de mensajes en clave y del diseño de criptogramas, y el *criptoanálisis*, que trata de hackear los mensajes en clave, rompiendo el criptograma. Gran parte de la población mundial utiliza a diario criptogramas con aplicaciones como WhatsApp que envía sus mensajes cifrados.

La *seguridad informática* es la rama de la informática que se ocupa de preservar la autenticidad, confidencialidad, disponibilidad e integridad de la información.

Este apéndice presenta una breve introducción a la criptografía y la seguridad informática como herramientas para comprender ciertos procedimientos descriptos en este libro.

A.1 Criptografía

Un criptograma está formado por dos algoritmos, uno que transforma un texto claro y una clave en un texto cifrado y otro que transforma el texto cifrado y otra clave (pueden ser la misma clave) en el texto claro nuevamente. La seguridad de los criptogramas se sustenta en algoritmos robustos y en claves suficientemente largas, jamás en el secreto del algoritmo. Los algoritmos robustos son aquellos a los que no se les conoce ataque teórico a los mecanismos criptográficos usados y las claves largas evitan ataques de fuerza bruta. Se denomina ataque de fuerza bruta a la prueba de todo el espacio de claves hasta encontrar la clave correcta.

Romper la seguridad de un sistema criptográfico robusto atacando la criptografía subyacente es muy dificil ya que supone resolver algún problema abierto de la matemática o encontrar soluciones alternativas de menor complejidad computacional a soluciones ya conocidas. Esto está reservado a expertos en criptografía. La máquina Enigma, el principal

dispositivo de cifrado usado por Alemania en la Segunda Guerra Mundial, fue hackeada de esta manera por el trabajo contínuo y sostenido de matemáticos cono M. Rejewski o A. Turing.

Atacar la implementación es más simple. Los programas fallan por motivos múltiples: código fuente incorrecto, errores en el compilador, sistema operativo mal configurado, o incluso vulnerabilidades en el hardware. Todas las implementaciones tienen vulnerabilidades que se pueden aprovechar. Algunas vulnerabilidades se introducen de forma deliberada para proporcionar puertas traseras. Estos ataques se llevan a cabo con especialistas en programación, electrónica digital, sistemas operativos y redes de comunicaciones. Cellebrite Mobile Synchronization y Micro Systemation AB son compañias que explotan vulnerabilidades de implementación para acceder con propósitos forenses a smartphones sin el PIN, la huella digital o el rostro esperado por el dispositivo.

Atacar con metodologías de ingeniería social es mucho más simple. Estos ataques están basados principalmente en el engaño. De nada sirve tener una contraseña secreta si se la revela a un supuesto administrador que la solicita por teléfono.

Criptograma

Es una cuaterna A, K, C y D donde:

a) A es un conjunto finito llamado alfabeto. Con sus elementos y respetando ciertas normas semánticas y sintácticas se generan el texto plano y el texto cifrado

b) K es un conjunto finito llamado espacio de claves. Sus elementos son todas las claves posibles k_i, tanto de cifrado como de descifrado

c) C es una función que toma elementos de A y devuelve elementos de A llamada algoritmo de cifrado

d) D es una función que toma elementos de A y devuelve elementos de A llamada algoritmo de descifrado

e) $C_{k1}(m)$ significa cifrar el texto plano m con el algoritmo C y la clave k_1

f) $D_{k2}(z)$ significa descifrar el texto cifrado z con el algoritmo D y la clave k_2

Algunos autores consideran dos alfabetos diferentes para construir los textos planos y cifrados, pero lo habitual es que sean el mismo. Es obvio que al descifrar un mensaje previamente cifrado se obtiene nuevamente el texto plano, o sea que

$$D_{k2}(C_{k1}(m)) = m \tag{A.1}$$

A.1.1 Criptografía Simétrica

La criptografía simétrica comprende aquellos criptogramas en donde la clave de cifrado k_1 y de descifrado k_2 son la misma clave o pueden deducirse una de la otra. El principal problema en comunicaciones de este tipo en donde la clave k debe ser conocida tanto por el emisor como en el receptor es el envío de esta clave de forma segura.

El algoritmo AES

Es un criptograma simétrico de cifrado de bloques diseñado para procesar 128 bits de datos con claves de hasta 256 bits. Es el algoritmo simétrico más popular y difundido. Fue presentado en un concurso en 1998 por J. Daemen y V. Rijmen y adoptado por el NIST en 2001 como estandar del gobierno de EEUU.

Descripción del algorimo

Cada bloque de 128 bits se organiza en una matriz cuadrada de 16 elementos de un byte cada uno. El algorimo repite en múltiples fases mecanismos de permutación y sustitución a los elementos de la matriz de las siguientes maneras:

a) Sustitución no lineal en donde cada byte de la matriz es reemplazado por otro de acuerdo a una tabla de búsqueda. Las tarjetas de coordenadas de los clientes de algunas entidades bancarias son un ejemplo conocido de estas tablas de búsqueda.

b) Transposición en donde las filas son desplazadas ciclicamente una determinada candidad de veces.

c) Sustitución que modifica cada byte de una columna en función de todos los bytes de la columna.

d) Operaciones XOR entre todos los elementos de la matriz y la clave expandida.

La implementación de estos cuatro mecanismos es muy simple, tanto en hardware como en software. Las sustituciones y transposiciones son obviamente reversibles y la XOR también lo es en virtud de que $A \oplus B \oplus B = A$. El algoritmo de descifrado es el mismo que el de cifrado como consecuencia de esta reversibilidad.

Consideraciones prácticas

No se ha confirmado ningún ataque exitoso contra el AES.

Un artículo de A. Shamir de 2015 presenta un ataque de canal lateral al AES basado en el análisis de los patrones de acceso de la memorias caches de los procesadores que ejecutan los algoritmos criptográficos. Esto es un ataque a la implementación y no al algoritmo.

A.1.2 Criptografía Asimétrica

La criptografía asimétrica abarca aquellos criptogramas en donde las clave de cifrado y de descifrado son diferentes, se generan a pares y no pueden deducirse una de la otra. Esto es algo relativamente nuevo, y al contrario de la criptografía simétrica que tiene miles de años, su existencia fue propuesta por primera vez por Diffie y Hellman en 1973. Un criptograma asimétrico queda completamente definido con tres algoritmos: el de cifrado, el de descifrado y el de generación de las claves.

El algoritmo RSA

Es un criptograma que utiliza la exponenciación modular para cifrar y descifrar. Su seguridad se basa en la complejidad computacional del problema de la descomposición de un número entero en sus factores primos. Es el criptograma asimétrico más aceptado e implementado a nivel global. Fue desarrollado por R. Rivest, A. Shamir y L. Adleman en 1979 y pertenece a los denominados algoritmos de flujo en donde cada caracter del texto plano se procesa uno por vez en una secuencia incremental de cifrado. Sujeto a controversias, no está probada ni rebatida su seguridad mientras las complejidades de los problemas de factorización de enteros o el cálculo de logaritmos discretos se consideren problemas abiertos de la matemática.

Generación de las claves

La secuencia de pasos para generar las claves es la siguiente:

a) Elegir dos números primos grandes p y q del orden de 10^{200} y calcular n y $\phi(n)$:

$$n = p \,.\, q$$

$$\phi(n) = (p-1).(q-1)$$

b) Seleccionar un entero e relativamente primo a $\phi(n)$, lo que significa que el máximo común divisor de e y $\phi(n)$ debe ser 1.

c) Seleccionar un entero d tal que $d.e \equiv 1 \; mod \phi(n)$. Esto equivale a decir que el resto del cociente entre $d.e$ y $\phi(n)$ debe ser 1

d) Finalmente se obtienen ambas claves k_1 y k_2 donde k_1 es (d,n) y k_2 es (e,n). Dado e no es posible obtener n y visceversa, lo que confirma que es un algoritmo asimétrico.

Cifrado y descifrado

Se tiene el texto claro M y ambas claves $k_1 = (d,n)$ y $k_2 = (e,n)$. Los textos claro y cifrado son en realidad secuencias de números enteros dado que RSA es un algorimo

de flujo en donde procesa secuencialmente números enteros de manera individual. El texto cifrado C se calcula con la siguente ecuación:

$$C = M^e \ mod \ n$$

En palabras, se toma el número M correspondiente al texto claro y se efectua la exponenciación M elevado a la e, el resultado se divide por n y el resto C de este cociente es el texto encriptado. El algoritmo de descifrado es el mismo que el de cifrado:

$$M = C^d \ mod \ n$$

Una característica notable del RSA es que si se cifra con la clave k_1 se descifra con k_2 y visceversa si se cifra con k_2 se descifra con k_1

Consideraciones prácticas

Luego de generar ambas claves, se difunde una de ellas haciéndola pública mientras la otra clave se mantiene secreta o privada.

Un mensaje cifrado con la clave privada de un individuo se considera auténtico, en otras palabras se dice que está firmado. Solo puede descifrarce con la clave pública de esa persona.

El procedimiento ideal para distribuir una clave pública es enviarla firmada por una entidad de certificación de la cual se conoce con certeza su clave pública. Esta autoridad de certificación hace las veces de notario.

Si se necesita enviar un mensaje autentico, íntegro y confidencial solo basta con cifrarlo dos veces, con la clave privada del emisor y con la clave pública del receptor. Solo el receptor podrá descifrarlo y además estará seguro de quién es el transmisor.

El RSA es bastante lento comparado con cualquier algoritmo simétrico y este es su principal inconveniente. En la práctica no se utiliza para cifrar grandes volúmenes de datos, pero es muy util para cifrar claves simétricas de sesión de algoritmos simétricos. Debido a esta lentitud, para firmar documentos se prefiere primero hacer un hash del documento y luego firmar el hash.

No se ha confirmado ningún ataque exitoso contra el RSA.

A.2 Funciones hash

Una *función hash* $H(x)$, a veces llamada *función resumen* es una función determinista que proyecta un conjunto D con infinitos elementos sobre un conjunto I con una cantidad finita de elementos:

$$\begin{array}{rccc} H: & \mathbb{D} & \longrightarrow & \mathbb{I} \\ & x & \longrightarrow & H(x) \end{array}$$

Las funciones hash son funciones sobreyectivas aplicadas sobre todo el conjunto imagen. Se dice que se produce una *colisión* cuando dos elementos distintos de D producen el

mismo hash. La existencia de colisiones es intrínseca a la definición de función hash debido a que la cardinalidad del conjunto D es mayor que la de I

Funciones hash criptográficamente seguras

Los procedimientos que aseguran y comprueban la integridad y la autenticidad de la información utilizan funciones hash con las siguientes propiedades:

a) $H(x)$ se aplica a bloques de datos x de cualquier tamaño

b) $H(x)$ produce una salida h de longitud fija

c) El cálculo de $H(x)$ es computacionalmente simple para cualquier x. Esto resulta en una sencilla implementación tanto en hardware como en software

d) Dado un x cualquiera, es computacionalmente intratable encontrar un $y \neq x$ en donde $H(y) = H(x)$

e) Dado un h cualquiera, es computacionalmente intratable encontrar un x tal que $h = H(x)$

f) Es computacionalmente intratable encontrar una pareja $y \neq x$ tal que $H(x) = H(y)$

Las funciones hash son ideales para averiguar si dos archivos son idénticos.Comparar resúmenes es más simple que comparar archivos completos, principalmente para archivos muy grandes. El algoritmo SHA-1 es un ejemplo de función hash que verifica estas seis propiedades. Admite entradas de hasta 2^{64} bits y produce un resultado de 160 bits.

Dos textos casi iguales producen resultados totalmente diferentes.

```
equipo:~$ echo "RSA es un algoritmo asimétrico" | sha1sum
83d991aa99b945a3d6e03803eceeb95eb6b07392
equipo:~$ echo "RSA es un algoritmo simétrico" | sha1sum
39a157f3391c552d35223197a56762cc5d760cda
```

A.3 Propiedades de la información

Las propiedades de la información que la seguridad informática debe preservar; y los hackers tratan de comprometer son la *autenticidad*, la *confidencialidad*, la *disponibilidad* y la *integridad*.

La información es *auténtica* cuando proviene de quien dice que proviene. Al recibir un mensaje es importante conocer de la identidad del emisor así como al instalar una

aplicación se debe tener la certeza de que es la aplicación deseada y no otra. Firmar un archivo es cifrar su hash con la clave privada de un algoritmo asimétrico.

La información es *confidencial* cuando acceden a ella solo las personas autorizadas. La confidencialidad se asegura cifrando la información con criptografía simétrica o asimétrica. Para enviar información con criptografía simétrica se necesita compartir la clave secreta con el problema que supone el envio de esta clave. Una solución a este inconveniente es enviar esta clave cifrándola dos veces con criptografía asimétrica, con la clave privada del transmisor y con la clave pública del receptor.

La información es *íntegra* si permanece inalterada. Las funciones hash son adecuadas para comprobar la integridad de archivos en virtud de que dos archivos casi idénticos producen hashes completamente diferentes.

La información está *disponible* cuando no se necesitan ni grandes recursos computacionales ni excesivo tiempo para acceder a ella. Obviamente, la disponibilidad es un concepto subjetivo. Los ataques contra la disponibilidad de la información se conocen como ataques de denegación de servicio.

A.4 Seguridad en iOS

iOS es un sistema operativo para dispositivos móviles de Apple que no permite su instalación en hardware de terceros. Es un sistema operativo like UNIX, deriva de macOS que a su vez está basado en Darwin BSD. Algunas funciones de seguridad, como el cifrado del sistema de archivos, no se pueden configurar para evitar que los usuarios las desactiven por error.

Arranque del sistema

Es un proceso de arranque seguro que garantiza que solo puedan ejecutarse apps y código de confianza.

Cuando se enciende el smartphone, el procesador de aplicaciones ejecuta código residente en una memoria de solo lectura grabada durante el proceso de fabricación del circuito integrado y, por lo tanto, implícitamente auténtica. Este sector de memoria contiene la clave pública de la entidad emisora de certificados de Apple, que se utiliza para verificar que el cargador de arranque iBoot esté firmado por Apple antes de permitir que se cargue. Este es el primer paso de una secuencia en la que cada paso garantiza que el siguiente está firmado por Apple. A continuación iBoot verifica y ejecuta el kernel de iOS. Si no es posible cargar iBoot o verificar el siguiente paso, el dispositivo entra en modo de recuperación y debe conectarse via USB al servicio en la nube iTunes para restaurar los valores de fábrica. El procesador de banda base procede de similar manera con software firmado y claves verificadas por Apple.

Apple incluye, además de los procesadores de aplicaciones y de banda base, el coprocesador Secure Enclave que utiliza un microkernel propio al que no se puede acceder desde el sistema operativo principal ni desde ninguna aplicación. El Secure Enclave cuenta con su espacio de memoria exclusivo y físicamente separado para almacenar, en otras cosas, claves privadas de 256 bits.

La representaciones matemáticas de las huellas digitales se guardan cifradas con una clave que está disponible solo para el procesador Secure Enclave. El Secure Enclave almacena huellas y comprueba si una huella digital coincide con los datos dactiloscópicos registrados. Estos datos no pueden accederse ni por el sistema operativo del smartphone ni por las aplicaciones. No se guardan en los servidores de Apple y nunca se hacen copias de seguridad en iCloud ni en ningún otro lugar. El Secure Enclave es también el también el responsable de procesar los datos del sensor Face ID para autenticación por la geometría del rostro. Las tres alternativas de autenticación que el sistema oparativo permite son la huella digital, la geometría del rostro y un código numérico.

Protección de los datos

La clave AES de 256 bits más importante en el proceso de protección de los datos de iOS es el device's unique ID (UID). Es único para cada dispositivo y lo genera el coprocesador criptográfico SEcure Enclave cuando se enciende el smartphone por primera vez. Ni el firmware ni el software pueden leerlo. El UID es utilizado exclusivamente por el AES engine, un circuito integrado de cifrado ubicado en el único bus que accede a la memoria donde se encuentra la partición de datos del usuario

Anteriormente, el UID se grababa en cada procesador durante el proceso de fabricación. En los procesadores T1, S2, S3 y A9 o posteriores de la serie A, cada Secure Enclave genera su propio UID. Puesto que el UID es único para cada dispositivo, y dado que se genera íntegramente en el Secure Enclave en lugar de generarse en un sistema de fabricación fuera del dispositivo, ni Apple ni ninguno de sus proveedores pueden acceder al UID o almacenarlo. Tampoco está disponioble mediante JTAG o alguna otra interfaz de depuración. El cifrado utilizando al UID como clave vincula los datos a un dispositivo determinado. No es posible acceder a los archivos si los circuitos integrados de memoria se trasladan físicamente de un smartphone a otro.

Los archivos de la partición de datos del usuario son cifrados por el AES engine con una clave diferente para cada uno de ellos. Estas claves se guardan en los metadatos de cada uno, y estos metadatos se cifran a su vez con la clave UID del sistema de archivos. Cuando necesita leer un archivo de la memoria flash, la AES engine descifra primero los metadatos y luego el archivo.

Protección de las aplicaciones y Jailbreaking

iOS solo ejecuta código firmado con un certificado emitido por Apple para garantizar que las apps provienen de una fuente conocida, aprobada y no se ha sido manipulada. A diferencia de otras plataformas móviles, iOS solo permite instalar apps procedentes de su tienda oficial. Durante la ejecución de una aplicación, se comprueba la firma digital de todas las páginas del código a medida que se cargan. Esto garantiza que la aplicación no ha sido modificada desde la última instalación o actualización.

La eliminación de estas medidas de seguridad impuestas por Apple se conoce como jailbreaking. Un dispositivo con jailbreak puede descargar aplicaciones, extensiones y temas que no están disponibles en la App Store oficial, conservando todas las demás funciones: la App Store, iTunes, llamadas telefónicas, etc. Desde julio de 2007, en donde se presentó el primer método de jailbreaking, se libra una carrera entre la comunidad de

hackers descubriendo vulnerabilidades y Apple reparándolas. Actualmente, la aplicación Electra 1131 permite hacer el jailbreak de dispositivos con versiones de iOS iguales o inferiores a la 11.3.1.

Jailbreak no es sinónimo de piratería, ya que tanto las herramientas para lograr el desbloqueo como las aplicaciones desarrolladas para funcionar en estos dispositivos no vulneran ningún derecho de propiedad. Apple afirma que el jailbreak anula la garantía, pero no tiene manera de hacer esto efectivo. Si fuera necesario usar el servicio técnico, se puede restaurar el smartphone a su estado original mediante iTunes y Apple no podría detectar que el dispostivo sufrió modificación previa alguna.

iCloud y la controversia de la seguridad de los datos en la nube

Apple afirma que su servicio iCloud guarda la información cifrada, pero los términos y condiciones del servicio sugieren lo contrario. Según Apple, los archivos se fragmentan y se guardan cifrados utilizando servicios de almacenamiento en la nube de terceros como Google Cloud Platform. Cada fragmento de archivo es cifrado con AES-128 y una clave derivada de una función hash del contenido del fragmento. Apple almacena las claves y los metadatos de los archivos en la cuenta de iCloud del usuario.

Las cláusulas de los términos y condiciones de iCloud establecen que Apple puede "revisar, mover, rechazar, modificar y/o eliminar contenido en cualquier momento" si el contenido se considera objetable o viola los términos del servicio. Además, Apple puede "acceder, usar, preservar y/o divulgar la información y el contenido de la cuenta a las autoridades policiales" siempre que lo exija o lo permita la ley. Apple también se reserva el derecho de revisar contenido que pueda violar los derechos de autor de los estatutos de la Digital Millennium Copyright Act (DMCA). Si los datos de iCloud estuvieran cifrados correctamente, Apple no podrían revisar el contenido, proporcionarlo a las autoridades o intentar identificar violaciones de derechos de autor

A.5 Seguridad en Android

Android utiliza el concepto de máquina virtual para la ejecución de las aplicaciones. Una máquina virtual es un software que simula ser una computadora y puede ejecutar programas como si fuera una máquina real. El lenguaje de programación Java popularizó esta idea al desarrollar la Java Virtual Machine (JVM) que es capaz de interpretar y ejecutar instrucciones expresadas en Java Bytecode generado por el compilador del lenguaje Java. Este código binario Java no es un lenguaje de alto nivel, sino un verdadero código de máquina de bajo nivel, incluso como lenguaje de entrada para un microprocesador físico. La JVM se encuentra en un nivel superior al hardware del sistema sobre el que se pretende ejecutar la aplicación y actúa como intermediaria entre el bytecode y el hardware. Toda aplicación Java es ejecutada en esta máquina virtual que convierte de código bytecode a código assembler del dispositivo final. La ventaja de las máquinas virtuales es aportar portabilidad, solo es necesario disponer de dicha máquina virtual para cada arquitectura. De ahí el famoso axioma: "escribirlo una vez, ejecutarlo en cualquier parte", o "Write once, run anywhere". Esto es muy importante en el mercado mundial de smartphones Android compartido por muchos fabricantes. Android utilizó una variante de máquina virtual java

llamada Dalvik hasta la versión 4.4.4. Dalvic interpretaba la mayoría del código de la aplicación pero realizaba la compilación y ejecución nativa de segmentos de código que se supone que se ejecutaban frecuentemente. Este concepto de localidad temporal del software ya es conocido y utilizado en memorias caches de instrucciones y en sistemas operativos, y en este contexto se denomina compilación just in time (JIT). La Dalvic Virtual Machine (DVM) carecía de la portabilidad de Java pero se obtenían aplicaciones con mejor rendimiento y menor consumo de energía. Estaba optimizada para requerir poca memoria y permitir ejecutar varias instancias de la máquina virtual simultáneamente, pero el sistema operativo era quien gestionaba la planificación de procesos, de la memoria y de los hilos. Dalvik siempre fue considerada una máquina virtual Java, pero esta no es una definición estrictamente correcta, ya que el bytecode no es Java Bytecode. Desde la versión 5.0 de Android fue sustituida por Android Runtime (ART) que compila el Java bytecode completo durante la instalación de una aplicación. Android Runtime introduce el concepto de compilación ahead of time (AOT), en donde se crea un archivo de compilación posterior a la instalación de cada aplicación. Esto evita que la aplicación se compile nuevamente cada vez que es ejecutada, reduciendo el uso del procesador y aumentando la duración de la batería al disminuir el número de compilaciones realizadas. Para mantener la compatibilidad con versiones anteriores, ART utiliza el mismo código de bytes de entrada que Dalvik, con archivos de extensión .dex como parte de los archivos APK, mientras que los archivos .odex son reemplazados por archivos de Formato Ejecutable y Enlazable (ELF). Una vez que una aplicación se compila utilizando ART, se utiliza exclusivamente el ejecutable ELF compilado. Este enfoque elimina diversos gastos involucrados con la compilación JIT, pero requiere de tiempo adicional para la compilación cuando se instala la aplicación. Además, las aplicaciones ocupan cantidades ligeramente mayores de espacio de almacenamiento para almacenar el código compilado. ART debutó como un entorno de ejecución alternativa en Android 4.4 KitKat y reemplazó completamente a Dalvik a partir de Lollipop.

Rooting y Flashing

Android utiliza un kernel like Linux y por lo tanto la mayoría de los conceptos presentes en Linux son aplicables a Android también. Ambos son sistemas operativos multitarea y multiusuario. Esto significa que pueden ejecutarse varios procesos simultáneamente y además que distintos usuarios pueden compartir el sistema sin conflictos entre si. Cada proceso se trata de manera independiente con un identificador de proceso asociado UID. Las aplicaciones tienen acceso a recursos limitados y se aplica el concepto de aislamiento de aplicaciones. El usuario root o superusuario es quien administra el sistema y tiene el poder de iniciar o detener cualquier servicio del sistema, editar o eliminar cualquier archivo y cambiar los privilegios de otros usuarios. Sin embargo, cuando una persona compra un teléfono Android, no se le permite iniciar sesión como usuario root de forma predeterminada.

Un dispositivo Android tiene varias particiones. El bootloader o cargador de arranque es un software ubicado en la primera partición que se ejecuta cuando se enciende el teléfono y configura el dispositivo a un estado inicial conocido. El trabajo principal de este gestor de arranque es montar las demás particiones y cargar el sistema operativo comúnmente

denominado ROM por defecto desde la partición Android. Los cargadores de arranque de Android son escritos por diferentes fabricantes y son específicos para el hardware sobre el cual se ejecutan. Para acceder al menú del cargador de arranque se requiere una particular combinación de teclas, como mantener pulsado el botón de encendido y pulsar el botón de subir volumen. Este menú ofrece diferentes opciones para arrancar desde otras particiones, y si se elige la partición de recuperación es posible, entre otras cosas, instalar actualizaciones del sistema operativo. Este es el modo estandar de instalar cualquier actualización oficial y autorizada del sistema operativo.

Rootear un teléfono Android es un anglicismo que se refiere a la obtención de permisos de acceso en el dispositivo para realizar acciones que normalmente no se permiten. Con el objetivo de minimizar las fallas es que los fabricantes de smartphones prefieren que sus dispositivos funcionen de una manera determinada para los usuarios normales. El rooteo de un dispositivo puede anular la garantía ya que root proporciona acceso abierto al dispositico y abre el sistema a vulnerabilidades al proporcionar al usuario capacidades de superusuario. Esto puede resultar muy peligroso si se instala una aplicación maliciosa. A pesar de que los fabricantes tratan de poner suficientes restricciones para evitar el acceso al usuario root, los hackers siempre han encontrado diferentes maneras de acceder. Los detalles de proceso de rooteo varían dependiendo del fabricante del dispositivo móvil, pero en lineas generales implica explotar un error de seguridad en el firmware del dispositivo para copiar el comando su a una ubicación en el path actual (/system/xbin/su) y conceder permiso de ejecución con el comando chmod. El rooteo es reversible, puede ser eliminado y el bootloader bloqueado de nuevo para acceder a la garantía.

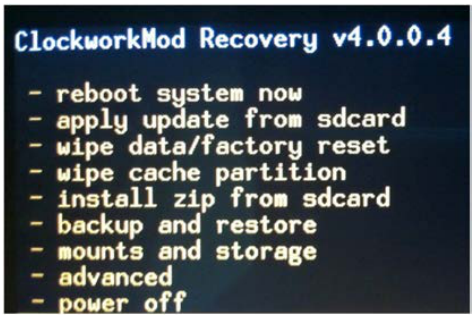

Figura A.1 Un programa de recuperación normal.

La instalación de un sistema operativo no autorizado por el fabricante se conoce como flashear el teléfono. Android es un software de código abierto que se distribuye bajo la licencia Apache V2, pero el bootloader casi nunca lo es. Como los fabricantes desean que sólo se instalen actualizaciones oficiales a través de la partición de recuperación, suministran un bootloader bloqueado, esto quiere decir que el dispositivo solo podrá ejecutar sistemas operativos aprobados por ellos. Esto quiere decir que la partición de arranque del sistema debe tener un firma digital conocida para ser considerada "booteable". Para instalar una versión de Android diferente en un dispositivo se debe primero desbloquear el bootloader o incluso reemplazarlo. El proceso de desbloqueo varía dependiendo del

dispositivo. Fabricantes como HTC, ponen a disposición de los usuarios en una sección para desarrolladores de su sitio web, instrucciones oficiales para desbloquear el bootloader de diferentes modelos de smartphones, por supuesto advirtiendo sobre la perdida de la garantía. El proceso de desbloquear el bootloader no debe ser confundido con el "rooteo", ni tampoco con el desbloqueo de la SIM (que permite usar el smartphone con cualquier operadora de telefonía del mundo que no sea la que lo vendió). Los programas de recuperación modificados proporcionan varias opciones que no se ven en el modo de recuperación normal. La figura A-01 muestra la captura de pantalla de un programa de recuperación normal. Solo se tiene un conjunto muy limitado de operaciones como hacer un reset de fábrica, instalar un sistema operativo o actualización oficial, limpiar la caché, realizar un backup de los datos de usuario o reinstalar algunos de estos backups. Esto puede variar ligeramente en función del fabricante pero las opciones nunca van más allá. La mayoría de los métodos de rooteo empiezan instalando un programa de recuperación modificado en la partición de recuperación. La figura A-02 muestra a Clockwork, un programa de recuperación muy usado. Es open source, gratis y distribuido bajo licencia Apache 2.0.

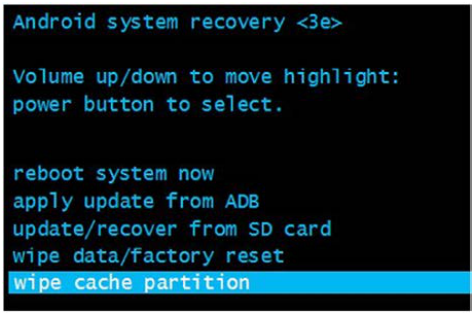

Figura A.2 Un programa de recuperación modificado.

Android Debug Bridge y Fastboot son dos herramientas muy utilizadas por desarrolladores Android y por aquellos que necesitan, por ejemplo, cambiar el sistema operativo o instalar un Recovery personalizado para rootear el sistema. Ambas funcionan desde la línea de comandos, característica que para algunos usuarios inexpertos pueden resultar complicado. Android Debug Bridge (ADB) permite interactuar con un smartphone desde una computadora. Es parte fundamental de Android Studio, el software para desarrollar aplicaciones en Android. A través de ADB se pueden ejecutar comandos para copiar archivos desde la computadora al teléfono, del teléfono a la computadora o reiniciar el dispositivo en el modo bootloader. ADB permite la comunicación con un smartphone encendido y con su sistema operativo Android funcionando.