Sistema de arranque UEFI: la seguridad, la adecuada instalación y la libertad de elección.

Fernando López, Lía Molinari, Claudia Banchoff, Javier Díaz

LINTI, Facultad de Informática, Universidad Nacional de la Plata
{flopez, lmolinari, cbanchoff, javierd}@info.unlp.edu.ar

Resumen. En el marco de modernizar el proceso de arranque de las computadoras y aportar una mayor seguridad al mismo, Unified EFI Forum propone la implementación de la interfaz UEFI (Unified Extensible Firmware Interface). La característica principal es que el firmware BIOS y el software utilizado estará firmado digitalmente. Este artículo analiza ventajas y desventajas de la utilización de UEFI, considerando entre otras situación la libertad de instalar diferentes sistemas operativos, a través de un caso de estudio, analizando su aplicabilidad.

Palabras claves: Gestor de arranque, sistema operativo, UEFI, GPL, BIOS

1 Introducción

Cuando se enciende una computadora se realizan una serie de pasos de los cuales el arranque del sistema operativo es trascendental. Esta acción se realiza a través de un gestor de arranque.

Si en la computadora conviven distintos sistemas o versiones de sistemas operativos, a través del gestor de arranque el usuario puede seleccionar cuál de dichos sistemas arrancará[1].

A continuación se describe el proceso de arranque (o "booteo") en computadoras basadas en IBM PC, ya sea con procesadores 16, 32 o 64 bits.

El BIOS (Basic Input Output System) es un software de bajo nivel que se halla en el motherboard. Cuando se arranca la computadora el BIOS se ejecuta, realizando el POST (Power-on self-test), que incluye rutinas que, entre otras actividades, fijan valores de las señales internas, y ejecutan test internos (RAM, el teclado, y otros dispositivos a través de los buses ISA y PCI). Luego se lee el primer sector del disco

[1] En este documento se usarán indistintamente los nombres x86 e IA32 para denominar a los procesadores de Intel de 32 bits y x86-64, x64 y AMD64 para denominar a la arquitectura de 64 bits desarrollada inicialmente por AMD y luego adoptada por Intel. Los procesos de arranque en equipos ARM e IA64 (Intel Itanium) están fuera del alcance de este documento.

de inicio (seleccionado de entre un conjunto de posibles dispositivos de arranque), llamado MBR (master boot record). Esto se carga en memoria y se ejecuta. El MBR puede contener un código de arranque denominado MBC (master boot code) y una marca de 2 bytes que indica su presencia o puede solamente contener la tabla de particiones. En el último caso el BIOS ignora este MBR [1].

Un gestor de arranque es un pequeño programa para arrancar un sistema operativo. Habitualmente se instala en el MBR y asume el rol de MBC.

En una computadora en la que hay sólo un sistema operativo, no hay referencias a pantalla generalmente. Si hay un gestor de arranque, este programa nos permitirá elegir el sistema operativo a arrancar.

El código del MBC de Windows, por ejemplo, busca en la tabla de particiones cual es la primer partición primaria con el flag de "booteo" activo y transfiere el control al código que se encuentra al comienzo de dicha partición: el PBR (partition boot record).

En el caso del sistema operativo Linux, se puede optar por distintos gestores de arranque, por ejemplo, LILO (Linux Loader), GRUB (Grand Unified Bootloader) o GAG (Gestor de arranque Gráfico).

LILO no se basa en un sistema de archivos específico. Funciona en una variedad de sistemas de archivos. GRUB en cambio, debe comprender el sistema de archivos y el formato de los directorios[2]. GRUB tiene algunas ventajas con respecto a LILO: tiene una línea de comandos interactiva como, permite arrancar desde una red, y podría considerarse más seguro [2].

El BIOS se utiliza en computadoras IBM PC y sus "clones" desde alrededor de la década del 80 con pocas modificaciones, siempre manteniendo compatibilidad hacia atrás a nivel binario. Esto provoca que el código de arranque que se almacena en el MBR y es ejecutado por el BIOS no pueda ocupar más de 440 bytes, deba ejecutarse en modo real x86 con direcciones de 20 bits (aprovechando segmentación), y utilizar interrupciones como medio de comunicación con el BIOS. Además, el BIOS está adaptado a arquitecturas x86.

Las restricciones de direcciones de memoria de 20 bits y el límite en el tamaño del gestor de arranque de 440 bytes no son aceptables en la actualidad considerando que los gestores de arranque modernos leen múltiples sistemas de archivos, muchas veces tienen interfaces gráficas con imágenes de fondo y tienen que cargar en memoria kernels en distintos formatos que ocupan mucho más de 220 bytes. Es por ello que habitualmente funcionan en múltiples etapas (o stages) y deben tener código para pasar a "modo protegido" en IA32 o "long mode" en AMD64, haciendo que el proceso de arranque sea mucho más complejo que lo que debería ser.

Por ejemplo el gestor de arranque GRUB 2, se divide en dos imágenes:

- "boot.img" es una imagen que se instala en el MBR. Esta imagen simplemente utiliza los servicios del BIOS para cargar una segunda imagen que tenga más funcionalidades. El BIOS no tiene soporte para leer sistemas de archivos (filesystems) así que boot.img tiene que saber en qué sectores se encuentra la siguiente imagen.

---

2 En las primeras distribuciones de Linux, se utilizaba LILO.

- “core.img” es una imagen que se limita en tamaño a 31KiB ya que su propósito es ser instalada en los sectores que quedan libres entre el MBR y el comienzo de la primer partición. Este espacio se denomina “MBR Gap”.

La imagen “core.img” cuenta con la capacidad de leer archivos desde el filesystem donde se encuentran los módulos y configuración de GRUB 2, y ya que salta a modo protegido supera la restricción de las direcciones de memoria de 20 bits. Además, tiene un modo “rescate” con una interfaz de comandos interactiva.

Instalar en el “MBR Gap” no es lo ideal ya que esta área de disco supuestamente no debe contener datos y los particionadores no siempre dejan este espacio libre. Sin embargo Grub y otras aplicaciones lo usan, ninguna con medios efectivos para detectar si otra aplicación lo está usando. Otra opción menos usada es reservar una partición sin formato que se conoce como BIOS Boot partition para instalar la imagen “core.img”. Esta opción suele usarse para instalar GRUB en sistemas que no soportan UEFI (la especificación que se tratará más adelante) pero que tienen un disco particionado con GPT, ya que GPT utiliza los primeros sectores del “MBR Gap” para almacenar su propia tabla de particiones [3].

2 Acerca de EFI

EFI (*Extensible Firmware Interface*), es una especificación que desarrolló Intel, que es un nexo entre el sistema operativo y el firmware. Desde este punto de vista puede verse como una alternativa para reemplazar la BIOS [4].

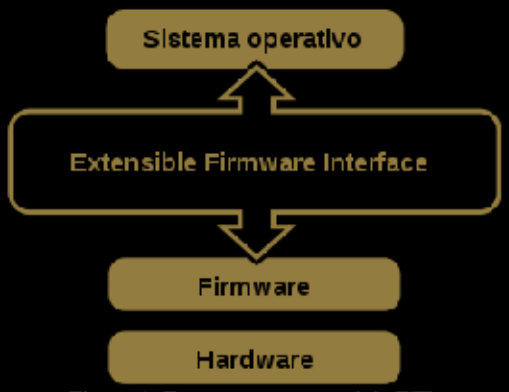

Figura 1: Esquema conceptual de EFI.

La EFI usa el sistema GPT (GUID Partition table) para solucionar algunas limitaciones del MBR, tales como la cantidad de particiones y capacidad máxima del dispositivo particionado.

GPT especifica la ubicación y formato de la tabla de particiones en un disco duro. Es parte de EFI. Puede verse como una sustitución del MBR como era pensado en la BIOS.

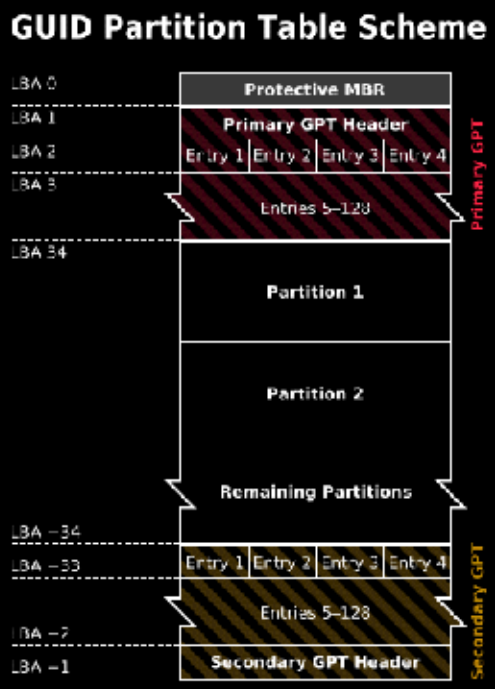

Figura 2: Esquema de particionado GPT.

Si bien se mantiene un MBR para tener compatibilidad con el esquema BIOS, la GPT en sí empieza en la cabecera de la tabla de particiones.

GPT usa modo de direccionamiento lógico (LBA, logical block addressing) en vez del modo cilindro-cabeza-sector usado con el MBR. El MBR "heredado" se almacena en el LBA 0.

En el LBA 1 está la cabecera GPT. La tabla de particiones en sí está en los bloques sucesivos. GPT proporciona asimismo redundancia. La cabecera GPT y la tabla de particiones están escritas tanto al principio como al final del disco lo cual provee redundancia, una importante característica del GPT.

UEFI Forum es una alianza entre varias compañías líderes de tecnología, con el objetivo de "modernizar" el proceso de arranque. Incluye representantes de AMD, American Megatrends, Apple, Dell, HP, IBM, Insyde Software, Intel, Lenovo, Microsoft y Phoenix Technologies. Continuó la evolución del EFI, creando la especificación UEFI (*Unified Extensible Firmware Interface*).

La especificación EFI es propiedad de Intel. La especificación UEFI es propiedad del UEFI Forum.

UEFI aporta criptografía, autenticación por red, y una interfaz gráfica.

2.1 Ambiente de pre-arranque (Preboot environment)

UEFI está definido para las arquitecturas IA32, IA64, AMD64 y ARM, pero para poder hacer una comparación objetiva con el BIOS tradicional detallaremos solamente el proceso de arranque de UEFI en las arquitecturas IA32 y AMD64.

La interfaz de comunicación entre el preboot environment y el sistema operativo, que define UEFI, consta simplemente de tablas de datos en memoria con información sobre el hardware y configuración del firmware. Estas estructuras tienen, además, punteros a las rutinas que implementan los servicios que el firmware ofrece a los bootloaders y a otras aplicaciones UEFI.

UEFI provee un Boot Manager que permite cargar aplicaciones y drivers UEFI desde el UEFI filesystem (este filesystem puede ser FAT12, FAT16 o FAT32) o por red. En este esquema el boot loader (por ejemplo la versión de Grub2 para UEFI) es un tipo especial de aplicación UEFI que al ser cargado se ejecuta ya en "modo protegido" o en "long mode" dependiendo de la arquitectura y tiene a su disposición los servicios de UEFI. De esta forma el boot loader no tiene las restricciones de tamaño impuestas por el antiguo BIOS ni la imposición de tener que arrancar en "modo real" (con direcciones de 20 bits).

A continuación, se enumeran las características y entorno de las aplicaciones UEFI:

- Las aplicaciones y drivers UEFI usan el formato Portable Executable PE32+, una modificación del formato PE/COFF usado habitualmente en los sistemas operativos de Microsoft.
- Procesador está en modo protegido o en "long mode"" al momento de ejecutar la aplicación.
- Acceso a UEFI filesystems a través de los servicios provistos por el firmware.
- Los servicios UEFI son expuestos en tablas de punteros.
- Los archivos en un UEFI Filesystem no pueden ocupar más de 4GiB por los límites impuestos por FAT 32.
- El UEFI Filesystem no puede ser más grande que 2TiB por los límites impuestos por FAT 32.
- Acceso a hardware adicional a través de drivers modulares en el UEFI filesystem o en las "option ROM" de dispositivos PCI.
- UEFI solamente exige que el firmware soporte binarios en la arquitectura del procesador y en EBC (EFI Byte Code), por lo que sucede que algunas implementaciones para procesadores AMD64 sólo pueden cargar, aplicaciones UEFI de 64bits y EBC a pesar de que estos procesadores tienen compatibilidad binaria con IA32.

2.2 EFI Byte Code Virtual Machine

UEFI define una máquina virtual que provee mecanismos independientes del procesador y de la plataforma para cargar y ejecutar drivers EFI. Esta máquina virtual se llama "EFI Byte Code Virtual Machine" (EBC Virtual Machine).

La EBC Virtual machine provee un conjunto de instrucciones en una arquitectura emulada de 64 bits (aunque la arquitectura real pueda ser de 32bits) con 8 registros de propósito general. El firmware del equipo debe implementar esta máquina virtual y poder cargar drivers en formato EBC desde la UEFI System Partition o desde una option ROM.

Es importante destacar que la versión 2.3.1_errata_B de la especificación UEFI especifica que las imágenes EBC deberían ser implementaciones de drivers, sin embargo nada parece impedir la implementación de otro tipo de imagen con este formato.

2.3 Proceso de arranque de UEFI en IA32 y AMD64

A continuación se describe el proceso de arranque normal de UEFI:

- Se busca una partición identificada por el código GUID "C12A7328-F81F-11D2-BA4B-00A0C93EC93B" en los dispositivos de almacenamiento particionados con GPT, o bien una partición con tipo 0xEF marcada como partición de arranque en dispositivos particionados con el esquema MBR. Esta es la "EFI System Partition" que debe tener formato UEFI filesystem.
- Si se encuentra esa partición, el firmware accede a su filesystem y carga el bootloader por defecto desde \EFI\BOOT\boot{arch}.efi o bien el bootloader o shell que tenga configurado el firmware.
- El bootloader es ejecutado en un entorno en "modo protegido" o en "long mode" dependiendo de la arquitectura del equipo.
- Utilizando los servicios provistos por el firmware UEFI, el bootloader puede cargar el sistema deseado, en caso que el kernel a cargar se encuentre en una partición con un UEFI filesystem. No es necesario que el bootloader tenga soporte para ningún filesystem ya que puede acceder al mismo usando los servicios de acceso a archivos provistos por el firmware.

A continuación se enuncian las características de la partición del sistema (System partition):

- Puede ser una partición en el MBR o en GPT.
- Puede ser un medio completo en el caso de pendrives y disqueteras.
- La partición estará formateada con un UEFI Filesystem (FAT).
- Debe tener un directorio \EFI, dentro del mismo cada "vendor" debe crear un directorio con un nombre que en lo posible no colisione con otros.
- Los medios extraíbles que sirvan como medio de arranque, como memorias USB deben contener un directorio \EFI\BOOT, dentro del mismo debe existir una imágen UEFI con el nombre boot{arquitectura}.efi. También es posible que el

firmware busque una imagen UEFI con este path en medios no extraíbles si no tiene configurada ninguna imagen en la NVRAM.

2.3.1 Grub2 para UEFI

Grub2 tiene una versión para instalar en sistemas UEFI. Normalmente se lo instala en \EFI\GRUB. En ese directorio se copia: el binario de Grub compilado como aplicación UEFI, el archivo de configuración grub.cfg y los módulos de Grub, que permiten, entre otras cosas, acceder a otros filesystems además del UEFI filesystem (a este último se puede acceder simplemente usando los servicios provistos por el firmware como se mencionó anteriormente).

3 Secure Boot

En las últimas revisiones de las especificaciones de UEFI se agregó la especificación de "Secure Boot" que propone mecanismos para tener un proceso de arranque seguro y libre de código malicioso. De ser bien implementados, estos mecanismos pueden ser aprovechados por cualquier sistema operativo libre o privativo para garantizar un arranque donde no sea posible la ejecución de código malicioso. Sin embargo si este mecanismo no es implementado de forma completa o correcta puede resultar muy restrictivo impidiendo la instalación de sistemas operativos que no sean de Microsoft, lo que afecta la libertad de optar por otros sistemas operativos.

Para validar que los drivers y aplicaciones UEFI cargadas no hayan sido reemplazados por código malicioso o incluso que no hayan sido alterados, UEFI propone utilizar pares de claves asimétricas, guardando en el firmware las claves públicas necesarias para verificar que cada ejecutable UEFI esté firmado por algún proveedor autorizado. La verificación de la firma fallará, si la clave privada con la que fue firmado el ejecutable fue revocada, está vencida, no es confiable o no corresponde con el ejecutable analizado porque éste fue alterado.

Los equipos que cumplan con la especificación de UEFI de Secure Boot deben contar con dos modos:

- Modo setup: en este modo el firmware no verifica que el gestor de arranque o los drivers estén firmados por una clave confiable. Además en este modo es posible registrar nuevas claves públicas sin ninguna autenticación usando la interfaz de usuario provista por el firmware.
- Modo usuario: en este modo solamente es posible cargar drivers y aplicaciones UEFI firmadas por claves confiables. Será fallido cualquier intento de cargar un gestor de arranque sin firmar.

Se utilizan dos clases distintas de pares de claves asimétricas:

- Key Exchange Keys (KEK): se utilizan para establecer una relación de confianza entre el sistema operativo y el firmware de la plataforma. Cada sistema operativo y cada aplicación de terceras partes que necesite comunicarse con el firmware debe tener instalada la parte pública de una KEK, en el firmware.

- Platatform Key (PK): su función es establecer una relación de confianza entre el dueño de la plataforma y el firmware. Para ello el dueño de la plataforma registra la parte pública de la PK en el firmware usando el servicio UEFI SetVariable(), y con la parte privada de la PK firma una o más KEKs, luego en el equipo (ya en modo usuario) se podrán instalar las KEKs firmadas.

Además de la PK y la base de datos de KEKs, el firmware debe almacenar una base de datos de firmas aceptadas (DB) y una base de datos de firmas no aceptadas (DBX). El firmware durante el proceso de arranque verifica que cada aplicación o driver UEFI tenga su firma registrada en la base DB y no en DBX. Si esto se cumple la aplicación o driver UEFI se ejecuta, sino el firmware no lo ejecutará a menos que tenga otros mecanismos no detallados en la especificación UEFI para validar el ejecutable [5].

Para modificar las bases DB y DBX hay que contar con la parte privada de la clave PK o de alguna KEK, también se pueden modificar estas bases de datos si el equipo está en modo setup.

Hasta que no se registre la PK, la plataforma opera en modo "setup". En este modo el firmware no debe requerir autenticación para modificar la PK o la base de datos de KEK. Luego de registrar la PK el firmware pasa automáticamente a modo usuario y queda en ese modo hasta que se borre la PK. Es posible verificar en que modo está el firmware leyendo la variable global UEFI "SetupMode".

Según el estándar UEFI es posible eliminar la PK registrada para volver al modo setup usando el servicio SetVariable() con una variable de tamaño 0 firmada con la parte privada de la PK. También se plantea otro posible mecanismo seguro dependiente de la plataforma, para eliminar la PK, en el cuál adicionalmente hay que modificar la variable "SetupMode" a 1. Por como está redactada la especificación este último mecanismo no parece ser obligatorio y no queda claro si cualquier usuario sin contar con la parte privada de la PK podrá acceder a este mecanismo. En el caso de los equipos que no sean ARM y cumplan con los requerimientos de certificación de hardware para sistemas cliente de Windows está especificado que este mecanismo dependiente de plataforma debe estar disponible para cualquier usuario físicamente presente.

3.1 Requerimientos de certificación de hardware para sistemas clientes y servidores Windows

La certificación de Microsoft Windows para sistemas clientes exige que estos equipos tengan Secure Boot habilitado por defecto. En el caso de los equipos que no tengan procesador ARM, deben existir dos modos de operación de Secure Boot:

- Standard mode: No plantea diferencias significativas con el modo usuario definido por UEFI. El firmware no presenta ninguna interfaz de usuario para modificar la PK, ni las bases de datos de KEKs, DB y DBX.
- Custom mode: Permite a un usuario con acceso físico al equipo cambiar todas las claves con una interfaz de usuario provista por el firmware e incluso eliminar la PK para pasar el equipo a modo setup.

- De esta forma en equipos IA32 y AMD64 que cumplan con esta certificación es posible pasar el equipo a modo setup usando una interfaz de usuario. Sin embargo en equipos ARM la certificación prohíbe el acceso al "custom mode" y al modo setup [6].

Una importante diferencia entre el estándar UEFI y la certificación de Windows, es que mientras el estándar plantea que quien controla la clave PK (y por lo tanto el software que podrá o no ser instalado el equipo) es el dueño del equipo, la certificación dice que el fabricante (original equipment manufacturer, OEM) será el único que tenga la parte privada de la PK. Esta es una diferencia fundamental: la libertad de elección del software a instalar en el caso del estándar la tiene el dueño del hardware, y, en el caso de la certificación, la tiene el OEM.

4 TianoCore

TianoCore es un portal que agrupa varios proyectos opensource relacionados con el desarrollo de firmwares y aplicaciones compatibles con UEFI. Entre estos se encuentra ***OVFM***, un firmware UEFI para el emulador QEMU, el virtualizador ***KVM***, actualmente siendo portado a XEN, y ***Duet***, una imagen de arranque para BIOS tradicionales que permite cargar un Shell UEFI en equipos que no soporten esta especificación. Con OVFM y Duet es posible probar UEFI en equipos que solamente tengan BIOS o que tengan versiones muy limitadas de este firmware.

Compilando OVFM con soporte para Secure Boot se puede ver una posible implementación de una pantalla para registrar y eliminar las claves y firmas necesarias de secure boot. Sin embargo, ya que OVFM + Qemu/KVM no permiten guardar los contenidos de la NVRAM entre reinicios, las pruebas que se pueden realizar son limitadas.

Es también interesante el uso de OVFM + Qemu/KVM como plataforma de pruebas para el desarrollo de distribuciones GNU/Linux que implementen UEFI.

Figura 3: OVFM con Secure Boot en modo Custom

Figura 4: Menú de OVFM para eliminar la PK actual o registrar una nueva.

5 Conclusiones

Uno de los tantos temores con esta especificación es si limita la instalación de determinados sistemas operativos, o dicho de otra manera, si sólo beneficia a las compañías que forman parte del Forum.

UEFI puede restringir al arranque de gestores que estén firmados con determinado certificado. Si bien se describe en la especificación la posibilidad de tener un modo setup que permite bootear sin verificar si el bootloader está firmado e incluso describe interfaces para agregar nuevas claves públicas, obviamente implementar estos mecanismos de forma completa y correcta es una elección de los fabricantes. En principio los dispositivos ARM que tengan la certificación de Windows no permitirán agregar otras claves públicas o entrar al modo setup. Al contrario los dispositivos IA32 y AMD64 que cumplan esta misma certificación deberán contar con mecanismos para habilitar el modo setup.

Por otro lado no es obligatorio que los Firmwares UEFI que se ejecutan sobre procesadores x86-64 soporten binarios de la arquitectura IA32, por lo que aún si se instala un sistema operativo de 32 bits en uno de esos equipos, el gestor de arranque debe ser un binario de 64 bits en un archivo PE32+, aún cuando los procesadores x86-64 pueden ejecutar código para procesadores IA32 sin problemas.

Si bien es posible escribir drivers UEFI independientes de la arquitectura usando EBC, el estándar sugiere no escribir otro tipo de imágenes UEFI (como gestores de arranque) que se compilen a EBC.

Las ventajas de UEFI sobre el BIOS tradicional de las IBM-PC para los desarrolladores de Boot Loaders es substancial respecto a que UEFI provee funcionalidades de alto nivel y rompe las limitaciones de uso de memoria y disco del esquema de arranque del BIOS.

En 2011 la Linux Foundation publicó el documento “Making UEFI Secure Boot Work With Open Platforms ”[7] en el cuál detalla como deberían implementar UEFI los fabricantes para permitir la instalación tanto de sistemas Open Source como sistemas operativos sin sacrificar los beneficios de Secure Boot. En el mismo se plantean al menos dos puntos disonantes con la certificación de Windows, por un lado la Linux Foundation recomienda que los equipos se vendan en modo setup (al contrario de la certificación de Windows que exige que estén en modo usuario), por otro lado se establece que en una primera instalación un sistema operativo que detecte que el firmware está en modo setup generará la parte pública y privada de la PK, guardando la parte privada en un medio a elección del usuario. La Linux Foundation, siendo fiel a la especificación UEFI y al contrario de la certificación de Windows delega el derecho a elegir en que sistemas confiar en el dueño de la plataforma.

Característica	Certificación de Windows	Recomendaciones de la Linux Foundation
Secure Boot	En modo usuario por defecto.	En modo setup por defecto.
Modo setup accesible	Sólo si el equipo no es ARM.	Siempre.
Control de la parte privada de la PK	El fabricante.	El dueño del equipo.

6 Referencias

1. Compaq Computer Corporation , Phoenix Technologies Ltd. , Intel Corporation .: BIOS Boot Specification Version 1.01. (1996)
2. Laurence Bonney.: Boot loader showdown: Getting to know LILO and GRUB. http://www.ibm.com/developerworks/library/l-bootload/index.html. (2005)
3. Free Software Foundation.: GNU GRUB Manual 2.00~rc1 http://www.gnu.org/software/grub/manual/html_node/BIOS-installation.html. (2012)
4. Unified EFI, Inc.: Unified Extensible Firmware Interface Specification Version 2.3.1, Errata B . (2012)
5. Intel.: Signing UEFI Applications and Drivers for UEFI Secure Boot - Revision 1.0. (2012)
6. Microsoft.: Windows Hardware Certification Requirements for Client and Server Systems. http://msdn.microsoft.com/es-ES/library/windows/hardware/jj128256. (2012)
7. Linux Foundation. http://www.linuxfoundation.org/publications/making-uefi-secure-boot-work-with-open-platforms . (2011)