



**Industria y Comercio**

**SUPERINTENDENCIA**

**Delegatura de propiedad Industrial**

**División de Nuevas Creaciones**

**SOLICITUD**

**PATENTE DE INVENCION**

21. **EXPEDIENTE No.** 05-51825

54. **TÍTULO** SISTEMA Y METODO PARA INICIO DE  
SISTEMA OPERATIVO PROTEGIDO UTILIZANDO VAUDACIOS  
DE LA CONDICION

51. **CLASIFICACIÓN INTERNACIONAL** G OGF 17/60

71. **SOLICITANTE** MICROSOFT CORPORATION

**DOMICILIO** REDMOND, WASHINGTON ESTADOS UNIDOS DE  
AMERICA

74. **APODERADO** DARLO CARDENAS NAUAS

22. **BOGOTÁ, D. C.,** 202099

**Industria y Comercio**  
**SUPERINTENDENCIA**

**SUPERINTENDENT** \_\_\_\_\_

participación : 40,00% 40,00%

POLICE: 14

REC'D: 10/11/65 11:00 AM

NUMBER 1 THE PHOENIX 2 1 NEWSPAPER 411 PRESTON  
LEFEBRE 1000 DIVISION 10 10000 100000

**FORMULARIO Unico de .....  
2000-3**

| SOLICITUD DE:                        |                       | PATENTE DE INVENCIÓN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |              | PATENTE DE MODELO DE UTILIDAD |      |
|--------------------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|-------------------------------|------|
| SOLICITANTE<br>(71)                  | Nombre:               | MICROSOFT CORPORATION                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |              | IDENTIFICACION                |      |
|                                      | Dirección:            | One Microsoft Way Redmond, Washington 98052, Estados Unidos de America                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |              | C.C.                          | NIT  |
| REPRESENTANTE<br>O APODERADO<br>(74) | Teléfono:             | 3123800                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Fax: 3122410 | C.E.                          | Otro |
|                                      | E-Mail:               | general@cardenasycardenas.com                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |              | Número                        |      |
|                                      | Lugar de Constitución | Estado de Washington, Estados Unidos de América                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
|                                      | Domicilio:            | REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
| INVENTOR<br>(ES)<br>(72)             | Nombre:               | DARIO CARDENAS NAVAS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |              | IDENTIFICACION                |      |
|                                      | Dirección:            | Carrera 7 No 71-52 Torre B Piso 9                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |              | C.C.                          | NIT  |
| DIRECCIÓN                            | Teléfono:             | 3123800                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Fax: 3122410 | C.E.                          | Otro |
|                                      | E-Mail:               | general@cardenasycardenas.com                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |              | Número 17.086.629 BTA         |      |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              | TP 1.250 C.S.J.               |      |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              | IDENTIFICACION                |      |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
| DOMICILIO                            | Nombre:               | 1. Bryan Mark Willman<br>2. Paul England<br>3. Kenneth D. Ray<br>4. Jalme Hunter<br>5. Lonny Dean McMichael<br>6. Derek Norman LaSalle<br>7. Pierre Jacomet<br>8. Mark Elliot Paley<br>9. Thekkthalackal Varugis Kurlen<br>10. David B. Cross                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |              | C.C.                          | NIT  |
|                                      |                       | 1. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>2. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>3. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>4. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>5. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>6. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>7. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>8. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>9. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America<br>10. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America |              | C.E.                          | Otro |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
|                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |                               |      |
|                                      | Teléfono:             | 3123800                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Fax: 3123800 |                               |      |
|                                      | E-Mail:               | general@cardenasycardenas.com                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |              |                               |      |
|                                      |                       | 1. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>2. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>3. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>4. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>5. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>6. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>7. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>8. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>9. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA<br>10. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA                                                                                                                                                                                                                                                 |              |                               |      |

● Título(54):

**SISTEMA Y MÉTODO PARA INICIO DE SISTEMA OPERATIVO PROTEGIDO UTILIZANDO VALIDACIÓN DE LA CONDICIÓN**

⑥

Prioridad

SI ☒

NO ☐

(33) País de Origen

USA

(32) Fecha

Junio 30, 2004

(31) N° de Solicitud

10/882,134

●

Para publicar a partir de la fecha de la presente solicitud a los:

☐

6 meses

☐

12 meses

☐

18 meses

☐

Otro

●

Comprobante de Pago No.:

05-38,821 / 05-38,824

Fecha

Mayo 26, 2005

**ANEXOS**

- ☒ Comprobante de pago de la tasa de presentación de la solicitud. (No. 05-38,821)
- ☐ Comprobante de pago de la tasa por concepto de excedente de palabras en la publicación.
- ☒ Comprobante de pago de la tasa por concepto de reivindicación de prioridad. (No. 05-38,824)
- ☒ Poderes, si fuere el caso. (Protocolo No 16.069)
- ☒ Certificado de existencia y representación legal cuando el solicitante sea persona jurídica. Protocolo No 16.069)
- ☒ Copia certificada de la primera solicitud, si se reivindica prioridad.
- ☒ Traducción simple de la primera solicitud, si se reivindica prioridad.
- ☒ Documento de cesión del Inventor al solicitante o a su causante.
- ☒ Descripción de la invención.
- ☒ Una o más reivindicaciones.
- ☒ Dibujos y/o planos necesarios.
- ☐ De ser el caso, copia del contrato de acceso.
- ☐ De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de la comunidad de las comunidades indígenas.
- ☐ De ser el caso, certificado de depósito del material biológico.
- ☒ Arte final 12x12 y 6x6 cm.
- ☒ Resumen y extracto para publicación.

10

NOMBRE:

DARIO CARDENAS/NAVAS

FIRMA:

C.C. 17.086.629 BTA

TP. 1.250  
C.SJ.

SUPERINTENDENCIA Y COMERCIO

NÚMERO DE VOUCHER: 00000000000000000000

VALOR: 1.000

FECHA VOUCHER: 26/05/2005

FOLIO: 1 DE 1  
RECEBIDO EN LA SUPERINTENDENCIA Y COMERCIO

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 05 - 38,821  
FECHA : MAYO 26 DE 2005

## \*\*\*\*\* CONSIGNACION \*\*\*\*\*

| DEPOSITANTE              | TIPO PAGO    | BANCO         | CUENTA      | No. PAGO | FECHA PAGO | Vr. PAGO     |
|--------------------------|--------------|---------------|-------------|----------|------------|--------------|
| CARDENAS Y CARDENAS P.I. | CONSIGNACION | BANCO POPULAR | 050-00110-6 | 36227317 | 26/05/2005 | 1,699,000.00 |

## \*\*\*\*\* CONCEPTO \*\*\*\*\*

| CANT. | RENTISTICO  | CONCEPTO                            | TOTAL CONCEPTO |
|-------|-------------|-------------------------------------|----------------|
| 1     | 50005-01-01 | SOLICITUDES                         |                |
|       |             | 1 TRAMITES DE SOL. DE PATENTE DE IN | 443,000.00     |
|       |             | TOTAL :                             | \$ 443,000.00  |

SON: CUATROCIENTOS CUARENTA Y TRES MIL PESOS

RESPONSABLE :

CARDENAS &amp; CARDENAS

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. SISTEMA Y MÉTODOS PARA EL INICIO DE SISTEMA OPERATIVO PROTEGIDO UTILIZANDO VALIDACIÓN DE LA CONDICIÓN.

ABORDAR

[illegible]

**NIT : 800176089-2**

\*\*\*\* CONSIGNACION \*\*\*\*

| DEPOSITANTE              | TIPO PAGO    | BANCO         | CUENTA      | No. PAGO | FECHA PGO  | Vr. PAGO     |
|--------------------------|--------------|---------------|-------------|----------|------------|--------------|
| CARDENAS Y CARDENAS P.I. | CONSIGNACION | BANCO POPULAR | 050-00110-6 | 36227317 | 26/05/2005 | 1,699,000.00 |

\*\*\*\*\* C O N C E P T O \*\*\*\*\*

| CANT. | RENTISTICO  | CONCEPTO                             | TOTAL CONCEPTO |
|-------|-------------|--------------------------------------|----------------|
| 1     | 50005-01-01 | SOLICITUDES                          |                |
|       |             | 82 INVOCACION DE UNA PRIORIDAD EN NU | 124,000.00     |
|       |             | TOTAL :                              | \$ 124,000.00  |

**RESPONSABLE :**

# CAMDENAS & PARTS

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. SISTEMA Y MÉTODOS PARA INICIO  
DE SISTEMA OPERATIVO PROTEGIDO UTILIZANDO  
VALIDACION DE LA CONDICION.

7

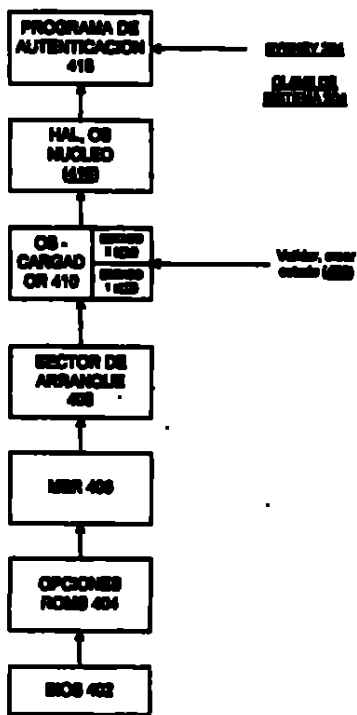


FIG. 1

ante para  
publicar

8

## **MÉTODO Y SISTEMA PARA EL INICIO DEL SISTEMA OPERATIVO PROTEGIDO USANDO VALIDACIÓN DE ESTADO**

### **CAMPO DE LA INVENCIÓN**

**[0001]** La invención presente generalmente se relaciona al campo de la computación. Más particularmente, la invención proporciona un mecanismo para asegurar que un sistema procede de un estado seguro conocido, y este mecanismo puede usarse para iniciar un sistema en cierto modo que proporciona seguridad suficiente del comportamiento correcto del sistema. Esta seguridad de comportamiento correcto, a su vez, puede impedir a uno o a más claves ser distribuidas bajo circunstancias inapropiadas.

### **EL FONDO DE LA INVENCIÓN**

**[0002]** La seguridad de la computadora es a menudo dependiente de poder predecir el comportamiento de componentes de software. En general, la seguridad de un sistema puede fluir desde la premisa de que un programa conocido cuyo comportamiento es conocido, del cual procede de un estado bueno, actuará de una manera predecible. Recíprocamente, la frustración de seguridad - qué puede involucrar conseguir un sistema de computadora para comportarse de maneras que están fuera de la contemplación de su diseñador - generalmente puede comprenderse como reemplazo o cambio de un programa conocido, o ejecutándose en un estado en que su conducta no se entiende. Así, un aspecto de mantener la seguridad de un ambiente de cómputo incluye verificar que un programa conocido se está usando, y que está procediendo de un estado bueno conocido.

**[0003]** Un área dónde la predicción de conducta es particularmente importante está en la carga de un sistema operativo y sus componentes. Aunque

el propio sistema operativo puede diseñarse para proporcionar algún nivel de fidelidad acerca de su comportamiento, el tiempo antes de que un sistema operativo se ha cargado es un tiempo donde el sistema es particularmente vulnerable al ataque, desde la infraestructura que protege al sistema operativo de los ataques puede no haber sido establecido todavía (o puede estar en proceso de establecerse). Así, asegurando que el sistema operativo carga de una manera predecible es importante proteger el sistema operativo de ciertas clases de ataques.

**[0004]** Un tipo de brecha de seguridad que puede fluir desde un cargue no seguro (non-secure loading) de un sistema operativo relacionado a la protección de la clave (o claves) que habilita cierta funcionalidad restringida. Por vía del ejemplo pero sin limitación, el sistema operativo MICROSOFT WINDOWS emplean una clave del sistema, o "SYSKEY" que se usa para proteger varios procesos haciendo la actuación correcta de procesos dependientes de la disponibilidad de SYSKEY. Por ejemplo, la clave necesaria para descifrar la información privada que es guardada por el sistema operativo en forma cifrada puede ser los derivada del SYSKEY.

**[0005]** Convencionalmente, las claves necesitadas realizar las operaciones restringidas son protegidas por el procedimiento de autenticación (logon). Típicamente, el usuario debe autenticarse correctamente (por ejemplo, proporcionando las credenciales de autenticación correctas, como una combinación de usuario/clave (username/password) antes de comenzar el uso del sistema. El uso de las claves sólo se habilita si el usuario se autentica correctamente, y el sistema le permitirá un número limitado de intentos a un sólo usuario (por ejemplo, tres) antes de concluir que el usuario ha fallado la autenticación propiamente. (Este tipo de límite en el número de intentos al autenticarse les impide a los usuarios desautorizados habilitar uso de funcionalidades protegidas usando un ataque de fuerza bruto para adivinar la

contraseña en el caso de, digamos, una computadora portátil robada.) Sin embargo, usando el procedimiento de autenticación para proteger el acceso a las claves asume que cargar el sistema operativo correctamente cargado con el programa de autenticación correcto, y que el uso de las claves no se ha habilitado por otra parte por código malicioso que puede estar corriendo. Si un cargador malicioso fuera usado en cambio, y el cargador malintencionado causa una autenticación del programa maliciosa para ser cargados con el sistema operativo, entonces el uso de claves podría habilitarse, o las claves incluso podrían divulgarse, habiendo entrado sin las credenciales correctas. Desde que la carga del sistema operativo proporciona una brecha de oportunidad de seguridad, la protección de las claves en tal una situación requiere que la carga del sistema operativo tome lugar bajo circunstancias dónde puede verificarse que tome lugar correctamente.

[0006] Un problema que ocurre con verificar la seguridad de un proceso de carga de sistema operativo es que la carga legítima del sistema operativo puede involucrar muchos programas diferentes (por ejemplo, hay numerosas "opciones diferentes de ROMs" que son pre-programas OS (sistema operativo) que corren durante un procedimiento de inicio de sistema), y hay numerosos procedimientos diferentes que pueden realizarse como parte de una carga de sistema operativo. Hay así, casi un incontable numero de estados de máquina legítimos diferentes durante una carga, e identifican tales estados y verifican que la máquina está en un estado bueno conocido que puede demostrar ser una tarea impracticable. Sin embargo, no todas las partes del procedimiento de carga tienen las implicaciones de seguridad. Puede ser más eficaz permitir que el cargue proceda sin ningún esfuerzo por evaluar su seguridad, pero luego configurar el ambiente a un estado bueno conocido antes de empezar cualquier procedimiento que podría afectar una función relacionada de seguridad, como la distribución de claves. Más generalmente, un sistema arbitrario puede permitirse correr durante algún tiempo sin cualquier tipo de evaluación de seguridad, con tal de que el sistema pueda

configurarse a un estado bueno conocido antes de permitir cualquier acción que puedan tener lugar implicaciones de seguridad.

**[0007]** En vista de lo anterior, hay una necesidad por un mecanismo que supere las desventajas del arte anterior.

## **RESUMEN DE LA INVENCION**

**[0008]** La invención presente proporciona el cargue de un sistema operativo bajo circunstancias dónde la carga puede verificarse para ser realizada correctamente. Cuando una máquina se inicia, el procedimiento de arranque (startup) muy temprano (por ejemplo, BIOS, opciones de ROMs, registro de inicio master, sector de arranque, etc.) son realizados. Después de que estos procedimientos tempranos se realizan, el cargador del sistema operativo se inicia, y puede realizar varias tareas preliminares. Después de que el cargador del sistema operativo se inicia y ha realizado tales tareas preliminares, una aprobación del cargador del sistema operativo se ha realizado.

**[0009]** La aprobación comprende las pruebas de validez de representación en el propio cargador, o una porción del cargador (ej., checksums, u otras pruebas que se apuntan a evaluar la identidad y exactitud del cargador), así como evaluar el estado de la máquina actual (u obligando la máquina a conformar un estado bueno conocido). Si el cargador (o la parte pertinente) se conoce para ser correcto, y el estado de la máquina es uno en que el cargador se ha verificado para comportarse correctamente previamente, entonces la conducta del cargador puede predecirse. Así, puede asegurarse que un cargador correcto que opera bajo un estado de la máquina correcto no cargará un componente que causaría datos que habilitan las funciones protegidas o (por ejemplo, las claves criptográficas, como la clave del sistema) para ser distribuida bajo circunstancias impropias.

**[0010]** Preferiblemente, la aprobación que se ha realizado por un validador que corre en un ambiente de alto-convicción. Un ambiente de convicción alto es aquel en que es posible proporcionar un grado relativamente alto de seguridad de que los procesos realizados en él correrán de la manera a que les esperan. Así, se asegura que el validador operará correctamente del hecho que el validador se verifica por un proceso que corre en el ambiente de alta-convicción, (por ejemplo, verificando una firma de su binario), y de la confianza subyacente que se procesa en el ambiente de alta-convicción que se llevará a cabo correctamente. (O, al menos hay un nivel de convicción con el cual el ambiente de alta-convicción no interferirá, o permite la interferencia correcta con la correcta operación de un proceso que opera dentro de tal medio ambiente, el funcionamiento correcto de un proceso que opera dentro de tal ambiente; uno todavía debe tener una base separada de confianza que la que el programa que lleva a cabo un proceso dentro del ambiente de alto-convicción realizará correctamente de la manera que se espera que realice.) Un ambiente de alta-seguridad puede proporcionar almacenamiento sellado, que es una facilidad del almacenamiento en que pueden sellarse los datos de un objeto particular (por ejemplo, a un programa particular), y que el control de ambiente de alta-seguridad es de tal manera que los datos sellados no se soltarán a cualquier objeto mas que el objeto al cual el dato es sellado (como es verificado por el ambiente de alta-seguridad). El validador puede usar este selló de almacenamiento para sellar una clave (por ejemplo, SYSKEY) a sí mismo, y puede negarse a abrir la clave excepto al objeto correcto, y cuando las circunstancias (por ejemplo, el estado de la máquina) es satisfactorio bajo algún estándar.

**[0011]** Otras características de la invención se describen debajo.

## **DESCRIPCIÓN BREVE DE LOS DIBUJOS**

**[0012]** El resumen anterior, así como la siguiente descripción detallada de personalizaciones preferidas, se entiende bien cuando es leída junto con los dibujos añadidos. Con el propósito de ilustrar la invención, allí se muestra en las construcciones de ejemplo de los dibujos de la invención; sin embargo, la invención no se limita a los métodos específicos y las utilidades expuestas. En los dibujos:

**[0013]** La FIG. 1 es un diagrama de bloque de un ejemplo de ambiente de cómputo en el cual pueden llevarse a cabo aspectos de la invención;

**[0014]** La FIG. 2 es un diagrama de bloque de un sistema que emplea un proceso cuyo funcionamiento correcto es dependiente de una clave del sistema;

**[0015]** La FIG. 3 es un diagrama de bloque de un sistema de archivo cifrado que protege los datos cifrados de ser descifrados sin autorización haciendo la descenccripción dependiente de una clave del sistema;

**[0016]** La FIG. 4 es un diagrama de flujo de un proceso de arranque de ejemplo con la aprobación de acuerdo con los aspectos de la invención;

**[0017]** La FIG. 5 es un diagrama de bloque de un validador del ejemplo de acuerdo con los aspectos de la invención; y

**[0018]** La FIG. 6 es un diagrama de flujo de un proceso del ejemplo de proteger una clave del sistema de acuerdo con los aspectos de la invención.

## **DESCRIPCIÓN DETALLADA DE PERSONALIZACIONES ILUSTRATIVAS**

### **Apreciación global**

**[0019]** Varios procesos que pueden tener lugar bajo un sistema operativo son dependientes de una o más claves para su operación correcto. El acceso a las claves puede ser controlado por un programa de autenticación, como un programa de logon que se niega a habilitar el uso de claves (key(s)) a menos que el usuario proporcione las credenciales correctas, como una combinación de usuario/clave. Así, por la negativa del programa de autenticación para habilitar el uso de claves en la ausencia de credenciales correctas, varios procesos (por ejemplo, descifrado de archivos cifrados) puede romperse (o completamente prevenirse) para usuarios que no conocen la contraseña. Mientras el programa de autenticación puede ser eficaz en el acceso de las claves, un cargador del sistema operativo podría engañarse en cargar un componente diferente que podría distribuir las claves fuera de las reglas de autenticación impuestas por el programa de autenticación. Así, cuando las claves son distribuidas de esta manera, mientras protege las claves requiere proteger el proceso cargante de sistema operativo. La invención presente proporciona mecanismos que pueden usarse para proteger el proceso cargante.

### **Arreglo de la Informática ejemplar**

**[0020]** La FIG. 1 muestras un ambiente de cómputo ejemplar en que pueden llevarse a cabo aspectos de la invención. El sistema del ambiente de cómputo 100 es sólo un ejemplo de un ambiente de informática conveniente y no se piensa sugerir ninguna limitación acerca del alcance del uso o funcionalidad de la invención. Tampoco debería ser el ambiente de cómputo 100 interpretado como para tener cualquier dependencia o requisito que se relaciona a cualquier

combinación de componentes ilustrados en el ambiente 100 de operación ejemplar.

**[0021]** La invención es operacional con numerosos otros propósitos generales o ambientes de sistema de cómputo o configuraciones especiales. Los ejemplos de sistemas de cómputo bien conocidos, ambientes, y/o configuraciones que pueden ser convenientes para el uso con la invención incluyen, pero no se limitan a, las computadoras personales, las computadoras del servidor, los dispositivos portátiles o portátiles, los sistemas de multiprocesador, los sistemas basados en microprocesador, las cajas de juego, la electrónica programable del consumidor, la red de PCs, miniordenadores, los sistemas informáticos grandes, los sistemas incluidos ("embebidos"), ambientes de cómputo distribuidos que incluyen cualquiera de los sistemas anteriores o dispositivos, y similares.

**[0022]** La invención puede describirse en el contexto general de instrucciones de computadora-ejecutables, como los módulos del programa, ejecutándose por una computadora. Generalmente, los módulos de programa incluyen las rutinas, programas, los objetos, los componentes, las estructuras de datos, etc. que realizan tareas particulares o implementan tipos de datos abstractos particulares. La invención también puede practicarse en ambientes de cómputo distribuidos dónde las tareas son realizadas por dispositivos de procesamiento remotos que se unen a través de una red de comunicaciones u otro medio de transmisión de datos. En un ambiente de cómputo distribuido, pueden localizarse módulos de programa y otros datos en los medios de comunicación de almacenamiento de computadoras locales y remotas incluso los dispositivos de almacenamiento de memoria.

**[0023]** Con referencia a la FIG. 1, un sistema ejemplar para llevar a cabo la invención incluye un dispositivo de cómputo de propósito general en la forma de una computadora 110. Los componentes de la computadora 110 pueden incluir,

pero no se limitan a, una unidad de proceso 120, un sistema de memoria 130, y un sistema de bus 121 que se acopla a varios componentes del sistema incluida la memoria del sistema de la unidad de proceso 120. La unidad de proceso 120 puede representar unidades de proceso lógicas múltiples como aquéllos apoyados en unos procesadores multi-hilos. El sistema de bus 121 puede ser cualquiera de varios tipos de estructuras de bus incluso un bus de memoria o controlador de memoria, un bus periférico, y un bus local que usan cualquiera de una variedad de arquitecturas de bus. Por vía de ejemplo, y sin limitación, tales arquitecturas incluyen el bus de la Industria de la Arquitectura Estándar (ISA), el bus de Arquitectura de Micro Canal (MCA), el bus ISA Extendido (EISA), el autobús local de la Asociación de Estándares de la Electrónica de Video (VESA), y el bus de Componente Periférico de Interconexión (PCI) (también conocido como el bus del Entresuelo). El sistema de bus 121 también puede implementarse como una conexión punto a punto, cambio de manufactura, o similares, entre los dispositivos de comunicación.

**[0024]** La computadora 110 típicamente incluye una variedad de medios legibles por computadora. Los medios legibles por computadora pueden ser cualquier medio de comunicación disponible que puede accederse por una computadora 110 y pueden ser incluidos ambos medios de comunicación volátil y no volátil, los medios de comunicación removibles y no-removibles. Por vía de ejemplo, y sin limitación, los medios legibles por computadora pueden comprender medios de comunicación de almacenamiento de computadora y medios de comunicación. Los medios de comunicación de almacenamiento de computadora incluyen ambos volátil y no volátil, los medios de comunicación removibles y no-removibles implementados en cualquier método o tecnología para el almacenamiento de información como las instrucciones legibles por computadora, las estructuras de datos, módulos de programa u otros datos. Los medios de comunicación de almacenamiento de computadora incluyen, pero no se limita a, RAM, ROM, EEPROM, memoria de destello (flash) u otra tecnología de memoria,

CDROM, los discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnetofónica, almacenamiento del disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que puede usarse para guardar la información deseada y que es accedida por la computadora 110. Los medios de comunicación incluyen típicamente las instrucciones leíbles por computadora, las estructuras de datos, los módulos de programa u otros datos en señales de datos moduladas como una onda portadora u otro mecanismo de transporte e incluyen cualquier medio de comunicación de entrega de información. El término "señal de datos modulada" es una señal que tiene una o más de un conjunto de características o cambia de tal manera para codificar la información en la señal. Por vía del ejemplo, y sin limitación, los medios de comunicación incluyen los medios de comunicación alambrados como una red alambrada o conexión directa-alambraron, y los medios de comunicación inalámbricos como el acústico, RF, infrarrojo y otros medios de comunicación inalámbricos. Las combinaciones de cualquiera de los anteriores también deben ser incluido dentro del alcance de los medios de comunicación leíbles por computadora.

[0025] El sistema de memoria 130 incluye los medios de comunicación de almacenamiento de computadora en forma volátil y/o memoria no volátil como la memoria de sólo lectura (ROM) 131 y la memoria de acceso aleatorio (RAM) 132. Un sistema básico de entrada/salida (input/output) (BIOS) 133, conteniendo las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 110, como durante el arranque (Start-up), que se guarda típicamente en ROM 131. La RAM 132 típicamente contiene los datos y/o módulos del programa que son inmediatamente accesibles y/o para ser operados presentemente por la unidad de proceso 120. Por vía de ejemplo, y sin limitación, la Fig. 1 ilustra el sistema operativo 134, el programa de aplicación 135, otros módulos de programas 136, y datos de programa 137.

**[0026]** La computadora 110 también puede incluir otros medios de almacenamiento de computadora removible/no-removible, volátil/no-volátil. Sólo, por vía de ejemplo la Fig. 1 ilustra una unidad de disco duro 141 que lee desde o escribe sobre los medios magnéticos no-removibles, no-volátiles, una unidad de disco magnético 151 que lee desde o escribe en un disco magnético 152 removible, no-volátil, y una unidad de disco óptica 155 que lee desde o escribe en un disco óptico 156 removible, no-volátil, como un CD ROM u otro medio óptico. Otros medios de almacenamiento de computadora removible/no-removible, volátil/no-volátil que pueden usarse en el ambiente de operación ejemplar incluyen, pero no se limitan a, los casetes de cinta magnetofónica, la tarjetas de memoria de destello (flash), discos versátiles digitales, cinta de video digital, RAM de estado sólida, ROM de estado sólido, y similares. La unidad de disco duro 141 se conecta típicamente al sistema de bus 121 a través de una memoria no-removible como una interfaz 140, y se conectan a la unidad de disco magnético 151 y la unidad de disco óptico 155 típicamente al sistema de bus 121 por una memoria removible, como la interfaz 150.

**[0027]** Las unidades y sus medios de almacenamiento de computadora asociados discutidos arriba e ilustrados en la Fig. 1, proporcionan almacenamiento de instrucciones leíbles por computadora, las estructuras de datos, los módulos de programa y otros datos para la computadora 110. En la Fig. 1, por ejemplo, la unidad de disco duro 141 se ilustra como almacén del sistema operativo 144, del programa de aplicación 145, de otros módulos de programa 146, y datos de programa 147. Note igual que estos componentes pueden ser diferentes del sistema operativo 134, el programa de aplicación 135, otros módulos de programa 136, y datos de programa 137. El sistema operativo 144, el programa de aplicación 145, otros módulos de programa 146, y datos de programa 147 tienen aquí números diferentes para ilustrar que, en un mínimo, ellos son copias diferentes. Un usuario puede entrar las órdenes e información en la computadora 20 a través de los dispositivos de entrada como un teclado 162 y dispositivo de

apuntamiento 161, normalmente llamado ratón, bola rotativa (trackball) o almohadilla de contacto. Otros dispositivos de entrada (no mostrados) puede incluir un micrófono, una palanca de mando, la almohadilla de juego, un plato satélite, escáner, o similares. Se conectan a menudo éstos y otros dispositivos de entrada a la unidad de proceso 120 a través de una interfaz de usuario 160 que se acopla al sistema de bus, pero puede conectarse por otra interfaz y estructuras de bus, como un puerto paralelo, un puerto de juego o un bus de serie universal (USB). Un monitor 191 u otro tipo de dispositivo de despliegue también se conecta al sistema de bus 121 vía una interfaz, como una interfaz de video 190. Además del monitor, las computadoras pueden incluir también otros dispositivos de rendimiento periféricos como parlantes 197 e impresora 196 que pueden conectarse a través de una interfaz periférica de salida 195.

[0028] La computadora 110 puede operar en un ambiente conectado a una red de computadoras que usa las conexiones lógicas a uno o más computadoras remotas, como una computadora remota 180. La computadora remota 180 puede ser una computadora personal, un servidor, un enrutador, una PC de red, un dispositivo par u otro nodo de la red común, y típicamente incluye muchos o todos los elementos descritos arriba relacionados a la computadora 110, aunque sólo un dispositivo de almacenamiento de memoria 181 se ha ilustrado en la Fig. 1. Las conexiones lógicas ilustradas en la Fig. 1 incluyen una red de área local (LAN) 171 y una red de área ancha (WAN) 173, pero también puede incluir otras redes. Tales ambientes de gestión de redes son comunes en las oficinas, en redes de computadoras de empresas grandes, intranets y la Internet.

[0029] Cuando se usa en un ambiente de red LAN, la computadora 110 se conecta a la LAN 171 a través de una interfaz o adaptador de red 170. Cuando es usado en un ambiente de gestión de redes WAN, la computadora 110 incluye un módem 172 u otros medios típicamente para establecer las comunicaciones sobre la WAN 173, como la Internet. El módem 172 que puede ser interno o externo

puede conectarse al sistema de bus 121 vía la interfaz de usuario 160, u otro mecanismo apropiado. En un ambiente conectado una red de computadoras, los módulos de programa representados relacionados a la computadora 110, o parte de eso, puede guardarse en el dispositivo de almacenamiento de memoria remoto. Por vía del ejemplo, y sin limitación, la Fig. 1 ilustra los programas de aplicación remota 185 como residentes en el dispositivo de memoria 181. Se apreciará que las conexiones de red mostradas son medios ejemplares y otros medios de establecer vínculos de comunicación entre las computadoras pueden ser usados.

### **Acción protegida por las claves**

**[0030]** Un ambiente de cómputo puede emplear una clave sobre la cual ciertos procesos que tienen lugar dentro del ambiente dependen para su funcionamiento correcto. La clave del sistema, o "SYSKEY" usado por los sistemas operativos MICROSOFT WINDOWS es un ejemplo de tal clave, pero no es un ejemplo limitando. En una personalización preferida, una clave sobre la que el proceso depende es única, criptográficamente clave aleatoria por-plataforma - es decir, dadas dos máquinas, es probable que las dos máquinas tengan claves diferentes. Así, un proceso que es dependiente de tales claves es improbable que sea portátil de una plataforma a otro - por lo menos en la magnitud que se emplean los mecanismos eficaces para asegurar que la clave de una plataforma no está disponible fuera de esa plataforma.

**[0031]** La Fig. 2 muestra un sistema de ejemplo en el cual corre un proceso que es dependiente de una clave. El proceso 202 es dependiente de la clave 204 para operar correctamente. Debe notarse que el proceso 202 no se limita a la noción tradicional de un proceso - es decir, una unidad de ejecución que puede manejarse por un sistema operativo y puede asignársele un espacio de dirección - sino que generalmente se refiere más a cualquier funcionamiento o serie de funcionamientos que pueden realizarse en una computadora. Debe notarse más

allá que, mientras este ejemplo muestra un proceso que es dependiente de una clave criptográfica, el término proceso como es usado aquí dentro no se limita a procesos que realizan las operaciones criptográficas.

**[0032]** Como es mostrado en la Fig. 2, si la clave 204 está disponible como una entrada de proceso 202, entonces el proceso 202 opera correctamente. Por otro lado, si la clave 204 no está disponible como una entrada de proceso 202, entonces el proceso 202 no opera correctamente. El mecanismo de protección 206 controla el acceso a la clave 204 - es decir, el mecanismo 206 proporciona cualquiera, o no proporciona, el proceso 202 con la clave 204, dependiendo de si las condiciones de seguridad pertinentes se han reunido. Por ejemplo, un usuario puede tener que autenticarse y proporcionar una contraseña correcta antes de que el mecanismo 206 habilite el uso de la clave 204.

**[0033]** Debe notarse que el rechazo de clave 204 para impedir que el proceso 202 opere correctamente a veces es un resultado deseado. Por ejemplo, el cifrado/descifrado (encryption/decryption) de archivos es un ejemplo de un proceso 202 que es protegido por la clave 204. El descifrar correctamente un archivo puede depender del acceso a la clave 204. Si el usuario no puede correctamente autenticarse, entonces puede ser deseable para el descifrado de un archivo no proceder, desde la incapacidad de un usuario para autenticarse puede indicar que la computadora está operándose por alguien diferente de su usuario intencional (por ejemplo, en el caso de una computadora portátil robada). Así, la clave de protección del mecanismo 206 puede hacer que el acceso a la clave 204 dependa de que las condiciones relevantes de seguridad sean reunidas, y puede usar el rechazo de la clave 204 para interrumpir procesos que necesitan ser interrumpidos cuando las condiciones de seguridad no se reúnen. La habilidad del mecanismo 206 para interrumpir los procesos de esta manera es dependiente de la clave 204 requiriendo de esos procesos para su funcionamiento correcto, desde

que esta dependencia habilita el acto de negar la clave 204 para interrumpir el proceso.

**[0034]** La Fig. 3 muestra un específico (pero no limitante) ejemplo de un proceso que es dependiente de la clave 204. En el ejemplo de la Fig. 3, el proceso del ejemplo es un Sistema de Archivos Encriptado (EFS) 302 que guarda los archivos en forma cifrada, y también descifra los archivos cifrados. Debe notarse que un propósito de tal encriptación de archivo es para proteger los datos en una computadora portátil de recuperarse por un ladrón en caso de que la computadora portátil sea robada. Cuando un archivo 304 se genera para el almacenamiento, el archivo se proporciona para EFS 302. EFS 302 cifra el archivo 304, y convierte el archivo 304 en un archivo 306 cifrado que se guarda en disco duro 141. Cuando una demanda se hace para recuperar el archivo 306 cifrado, EFS 302 recupera el archivo 306 cifrado y lo descifra para generar el archivo 308 descifrado. (En la práctica, el meollo del archivo 308 descifrado es igual que el archivo 304 original, aunque para claridad la Fig. 3 muestra estos dos casos de archivo separadamente: el archivo 304 es el archivo original, y el archivo descifrado 308 es el mismo archivo después de que se ha cifrado, guardado, recuperado, y descifrado por EFS 302.)

**[0035]** Se notará que una entrada a EFS 302 es la clave 310 satisfecha. La clave 310 satisfecha es preferiblemente una clave simétrica que funciona como una entrada a un método criptográfico. La clave 310 satisfecha se usa para cifrar el archivo 304 y crear el archivo 306 cifrado, y se usa para descifrar el archivo 306 también cifrado y crear el archivo 308 descifrado. Puede apreciarse que guardando la clave 310 en algún lugar recuperable pudieran dar EFS rápidamente a la habilidad de 302 para proteger los datos ineffectivamente: Si el contenido de la clave satisfecha está fácilmente disponible en un disco duro, o si es fácilmente determinable de algún rasgo conocido de la computadora portátil (por ejemplo, el número de serie del procesador), entonces daría lo mismo si los archivos se

guardaron en forma cifrada, desde que un ladrón pudiera encontrar la clave fácilmente y podría descifrarlos. Así, es deseable proteger la clave derivándolo de alguna manera que sólo puede hacerse con la cooperación del verdadero dueño. Una manera de proteger la clave es usar un generador importante módulo 312 que recibe la clave 204 como la entrada y deriva la clave 310 satisfecha como una función de la clave 204. Así, para extender que la clave 204 se proporcionará solo bajo las circunstancias apropiadamente seguras, es igualmente verdad que la clave 310 satisfecha será sólo derivable bajo las circunstancias apropiadas. En otros términos, haciendo la derivación de clave 310 satisfecha dependiente de la disponibilidad de la clave 204, cualquier protección que se permite para codificar la clave 204 puede extenderse al contenido de la clave 310. Por ejemplo, si proporcionar la clave 204 requiere que el usuario se autentique proporcionando una contraseña correcta, puede asegurarse que el contenido de la clave 310 no estará disponible a menos que el usuario se autentique correctamente.

[0036] Así, la protección de la clave 204 de ser dada bajo circunstancias erróneas es importante, desde que otros procesos pueden depender de la clave 204 que sólo se proporciona cuando el contexto de seguridad es correcto (por ejemplo, una legítimo, usuario autenticado que proporciona la contraseña correcta) está presente. Como es descrito abajo, una manera de causar que una máquina distribuya la clave 204 de una forma que puede producir mal uso es iniciar la máquina en un modo no seguro en el cual los componentes falsos del sistema operativo pueden reemplazar los componentes correctos (donde los componentes correctos se presumen para poder proteger la clave 204). Así, se describen los mecanismos debajo que aseguran que una máquina es iniciada en un ambiente conocido (seguro) antes de que la clave 204 puede distribuirse.

### **El Proceso de Inicio con la Validación de Estado**

**[0037]** La Fig. 4 muestra una sucesión típica de eventos que se usan en un procedimiento de inicio (booteo).

**[0038]** Inicialmente, la máquina es encendida. Una máquina típica se configura para empezar a ejecutar instrucciones en alguna dirección fija en el momento que es encendida. Las instrucciones típicamente contenidas en esta dirección son conocidas como "BIOS" 402, o "Sistema de entrada/salida (Input/Output) Básico". En la conclusión de ejecución de la BIOS 402, La BIOS 402 inicia la ejecución de pequeños programas llamados "opciones de ROM" 404. Las opciones de ROM son programas que realizan las funciones básicas de arranque temprano, tal como configurar la contraseña de hardware para la máquina, o seleccionar cuál de varios sistemas operativos será iniciado. Después de que la opción ROM 404 se ejecuta, la máquina es instruida para cargar el Registro Principal de Arranque (MBR – Master Boot Record) 406.

**[0039]** El MBR 406 es un programa ejecutable. Típicamente, el MBR 406 reside en el primer sector del disco duro de una computadora, y empieza por buscar en la tabla de particiones que partición debe usar para el arranque. (Por ejemplo, un disco puede dividirse para el uso con sistemas operativos diferentes, y cada sistema operativo puede requerir un procedimiento de arranque diferente.) Después de que la partición correcta se ha buscado, el MBR 406 transfiere el control al sector de arranque 408 asociado con esa partición. El sector de arranque entonces comienza el proceso de cargar el sistema operativo cargador 410 que cargará el sistema operativo finalmente. Debe notarse que el MBR 406 se muestra meramente en la Fig. 4 para ilustrar cómo tal componente encajaría en un proceso de arranque de ejemplo, y la invención no se limita a los procedimientos de arranque que usan MBR 406.

**[0040]** Durante el tiempo que el cargador del sistema operativo 410 se está ejecutando, el cargador y el estado de la máquina (450) son validados. (la

Máquina, en este contexto, puede referirse a una máquina física o una máquina virtual.) La validación se ha realizado por un componente de software confiable que opera en un ambiente de alta-seguridad (una personalización de cual es más particularmente descrita debajo), y por lo tanto hay algún grado de seguro/confiabilidad (assurance/trustworthiness) que la validación del cargador 410 es hecho correctamente. En esencia, para validar los aspectos pertinentes del estado de la máquina mientras el cargador del sistema operativo 410 tiene el control de la máquina, y valida el cargador 410, es posible proporcionar algún nivel de seguridad que ilegítima o falsea las versiones de los componentes del sistema operativo (por ejemplo, la capa de abstracción de hardware, el núcleo (kernel), los controladores, etc. (416)) no se cargará después, al tiempo que la autenticación del programan 418 es ejecutado. Prevenir que componentes falsos sean cargados al tiempo que la autenticación del programan 418 es ejecutada es importante porque uno de las primeras cosas que el sistema operativo hará después de que está cargado es ejecutar un programa de autenticación 418 que conduce el acceso de la clave 204, y si pudieran cargarse los componentes falsos, entonces estos componentes podrían causar que el programan 418 se comporte mal de una manera que podría producir distribución de la clave 204 bajo circunstancias impropias, por lo tanto comprometiendo la seguridad de todos los componentes de los que dependen de la protección de clave 204 como fue descrito anteriormente. Así, la protección de la clave 204 puede lograrse ejerciendo el control correcto sobre el estado de la máquina del tiempo desde que cargador del sistema operativo 410 se ejecuta hasta que la autenticación del programa se ha completado.

[0041] La validación 450 toma lugar durante el tiempo que el cargador del sistema operativo 410 se está ejecutando. La aprobación incluye la validación del cargador y el estado de la máquina, y también puede incluir la configuración del estado de la máquina a un estado bueno conocido. La idea básica detrás de validar y poner el estado de la máquina es poner la máquina en un estado dónde,

si el cargador se ejecuta mientras la máquina está en ese estado, entonces el cargador no cargará ningún componente falso o por otra parte se comportará de manera que producirían la brecha de seguridad. La validación 450 asegura que el código del cargador es, de hecho, el código que se ha verificado para comportarse correctamente previamente, y también asegura que la máquina está en un estado bajo el cual el código es conocido se comportará correctamente (verificando que la máquina ya está en ese estado, o poniendo la máquina en ese estado). Se apreciará que esta técnica puede apalancar el existente estado de seguridad de la máquina para asegurar que los estados futuros de la máquina también estarán seguros, y este apalancamiento es hecho posible por el hecho que el comportamiento del cargador del sistema operativo 410 no sólo es conocido y entendido, pero también herméticamente circunscrito. Basado en esta observación, se apreciará que la validación 450 no debe tomar demasiado temprano (por ejemplo, en el momento que la opción de ROM 404, o MBR 406, se ejecuta), desde la variedad amplia de código de las numerosas fuentes diferentes, y la inmensa anchura de estados que la máquina podría ponerse como resultado de ejecutar tal código, no lo hace difícil, si no imposible, para determinar el comportamiento de la máquina mientras se ejecutan todos estos varios módulos de código diferentes. Así, es preferible no tener relación con qué los estados de la máquina sean previos a la ejecución del cargador 410, con tal de que la máquina pueda ponerse en un estado legítimo en el momento que el cargador 410 se está corriendo.

[0042] En una personalización preferida, la ejecución del cargador 410 es dividida en dos estados: estado 1 (412) y estado 2 (414). Preferiblemente, antes de que la fase 2 entre, el código que implementa la fase 2 (450) se valida, y el código confiado que realiza la aprobación 450 entonces salta al estado 2 a un punto de entrada bien-definido. (El "código confiado" que realiza la aprobación es el programa antes dicho que corre en un ambiente de alta-seguridad. Se entenderá por aquéllos con habilidad en el arte que "confiado" no implica la

infalibilidad absoluta, pero meramente significa que hay alguna base para asumir que el código hará su trabajo correctamente. Desde que el comportamiento de un programa puede afectarse por el ambiente en que corre, mientras se ejecuta el código confiado en un medio ambiente de alta-seguridad el código confiado operará correctamente: (1) el código confiado puede confiarse para realizar su función correctamente en un ambiente que conforma algunas expectativas, y (2) en el ambiente de alta-seguridad puede confiarse para proporcionar un ambiente que conforma tales expectativas correctamente.) El estado 2 puede validar entonces (según algunos la norma definida) cualquier información que ha recibido el estado 1. La línea divisoria entre el estado 1 y el estado 2 refleja el hecho de que puede haber algunos aspectos de un procedimiento de arranque que pueden realizarse sin ninguna comprobación de seguridad, y sin tener ninguna consecuencia para alguna tarea de seguridad-relacionada (como la distribución de la clave 204), con tal de que puedan validarse los programas importantes y el estado en algún punto antes de que pudieran ponerse los eventos en movimiento que podría causar que una tarea de seguridad-relacionada no sea realizada correctamente. Más aun, puede haber problemas, con intentar validar el estado demasiado temprano en el proceso de arranque: la ruta de ejecución real que legitima el proceso de arranque que puede tomar es bastante inconstante, por eso es difícil definir la diferencia entre los estados de la máquina válidos o no válidos bajo tales circunstancias inconstantes. Puede tener sentido para permitir la carga simplemente proceder a través de cualquiera de estos estados inconstantes, sin intentar determinar si algún estado a través del cual la máquina pasa es válido. Así el estado 1 representa la porción de la carga que procede sin ninguna aprobación. Al final del estado 1, se validan el cargador y el estado de la máquina, y el estado 2 empieza. La presunción de una línea divisoria es que los procesos pueden ejecutarse sin ninguna restricción de seguridad, a un cierto punto - a tal punto se validan todos los factores pertinentes sobre la máquina y la máquina es puesta en un estado bueno conocido que esencialmente anula el efecto de cualquier acción anterior que podría haber puesto la máquina en un estado que es inaceptable

desde un punto de vista de seguridad. El punto exacto en el proceso de carga donde la línea divisoria entre el estado 1 y el estado 2 es arrastrada es muy específico a las circunstancias (por ejemplo, ¿Qué código del cargador se parece? ¿Qué tarea necesita ser hecha como parte de la carga?), y representa una clase de intercambio: Por un lado, la línea divisoria debe estar bastante tarde, para que las acciones futuras, la cantidad de variación en la conducta de carga legítima sea bastante pequeña que pueda distinguirse el comportamiento legítimo factiblemente de la conducta ilegítima. (Como se notó anteriormente, en el procedimiento de carga temprano, el número grande de opción que las ROMs y otras variables hacen el número de posibles caminos de ejecución tan grande que es difícil distinguir las conductas legítimas de las conductas ilegítimas.). Por otro lado, la línea divisoria debe estar bastante temprana que preceda cualquier evento (por ejemplo, la carga del programa de autenticación) eso podría tener un efecto en la seguridad, como causar que las claves sean distribuidas inapropiadamente. En general, la línea divisoria entre el estado 1 y el estado 2 permite primero ejecutar un sistema arbitrario en un "raro" (o "abierto", o "inválido") estado durante algún tiempo, y entonces el inicio de cualquiera componente es necesarios para verificar el comportamiento (por ejemplo, un ambiente de alta-seguridad), y entonces usa tales componentes para validar el estado actual de la máquina (o fuerza la máquina a un estado bueno conocido) a tal punto que el proceso de hacer algo puede afectar la seguridad después (por ejemplo, la distribución de claves) puede permitirse proceder.

### **Validación 450**

[0043] La validación 450 es, en esencia, el acto de verificar que el sistema operativo cargador 410 (o, más específicamente, en una personalización preferida, verificar ese estado 2 del cargador) es un programa confiado conocido, y asegura que la máquina en que está corriendo está en un estado bueno conocido. Así, la aprobación tiene dos partes: (1) examinar el programa del cargador (o parte 2 del

cargador de programan) para asegurar que es el conocido, el programa confiado que se cree que es, y (2) cambiar el estado de la máquina pertinente a un estado bueno conocido bajo el cual el cargador se conoce que se comporta correctamente. La premisa de realizar (1) y (2) es que un programa conocido, operando en un estado de la máquina conocido, se comportará de una manera conocida.

[0044] La primera parte de la validación - es decir, examinar el cargador - puede realizarse en una variedad de maneras. En el ejemplo más simple, puede haber un detalle del programa conocido (qué pueden ser criptográficamente firmados para crear una firma digital, o sino almacenados de alguna manera no-engañada) contra la cual el programa real que está corriendo puede compararse. El problema con tomar un detalle sobre el programa completo es que casos diferentes de un programa legítimo pueden tener imágenes ligeramente diferentes, para que un detalle que se calcula que en cierto modo requiere la identidad completa entre el programa conocido y el caso corriente del programa puede ser demasiado restrictivo. Preferiblemente, la aprobación se ha realizado en cierto modo que asegura que el programa es lo que se supone que es, pero no es demasiado restrictivo - por ejemplo, el proceso de aprobación puede comparar detalles de componentes del programa que se conocen para ser arreglados, y puede realizar otras pruebas en componentes del programa que pueden cambiar con el tiempo. El componente que realiza la aprobación debe detallarse para las especificidades del programa que está siendo validada; la invención no se limita a cualquier técnica particular para validar un programa.

[0045] La segunda parte de la aprobación - es decir, verificar/configuración (verifying/setting) del estado de la máquina - se realiza preferentemente poniendo todas las "fuentes" pertinentes de estado a los valores conocidos. Típicamente, el estado pertinente que puede afectar un programa que opera viene de tres fuentes: la CPU, el chip de configuración (chipset), y la memoria. Así, estos artículos

pueden ponerse en un estado bueno conocido en el momento de aprobación - por ejemplo, la CPU puede ponerse en un estado conocido - por ejemplo, anillo 0, con el contador del programa apuntado a una situación conocida, que todo los datos se registran a cero, y toda la memoria de otra manera que la memoria en que el programa del cargador se almacena para poner a cero. Si el cargador se ha verificado para comportarse correctamente cuando corre con la máquina en este estado, entonces la combinación de examinar al cargador y poner la máquina en el estado bueno conocido debe asegurar la conducta correcta al tiempo en que el programa de autenticación 418 se ejecuta. Como se discutió anteriormente, uno de los beneficios (aunque no el único beneficio) del comportamiento correcto del funcionamiento del programa de autenticación 418 es que los componentes falsos no se cargarán porque causaría que la clave 204 sea distribuido bajo circunstancias impropias.

[0046] Debe notarse que, para que el proceso de aprobación proteja la carga del sistema operativo de los componentes falsos, debe haber convicción suficiente para que el proceso propio de aprobación este realizándose correctamente. La exactitud del proceso de aprobación puede asegurarse incluyendo el validador como un agente confiado que corre sobre o en un ambiente de cómputo de alta-seguridad. Puede haber por ejemplo, un pequeño, sistema operativo que realiza un conjunto limitado de funciones, pero que proporciona un alto grado de convicción tal, que un sistema operativo realizará según sus especificaciones. Tal sistema operativo podría ejecutarse junto con otros sistemas operativos en una sola máquina, y la independencia de tal sistema operativo de alta seguridad de otros ambientes (menos seguro) en el sistema podrian ser forzados por un componente de supervisión, tal como un supervisor o monitor de máquina virtual. (En una personalización, un supervisor da fuerza a particiones que son ambientes que el supervisor mantiene en algún estado de aislamiento mutuo de nosotros y en que un sistema operativo puede correr.) Es más, el componente de seguridad alta puede tener acceso exclusivo a una raíz de

confianza, como un módulo de hardware que aplica, y fuertemente puede proteger, las claves criptográficas por-plataforma. El validador podría ser un programa (o "agente") que corre en tal ambiente de alta seguridad que proporcionaría la seguridad de la magnitud que el propio ambiente de alta-seguridad puede confiarse para resistir los ataques - el validador no estaría sujeto a los ataques de afuera del ambiente de alta-seguridad que causaría que se comportase incorrectamente. Adicionalmente, como es notado sobre el ambiente de alta-seguridad podría proporcionar el almacenamiento sellado (es decir, la habilidad de guardar algún dato y sólo soltar ese dato a un objeto particular para el cual el dato se ha sellado), y el validador podría usar este almacenamiento sellado para guardar cualquier clave que será distribuida. Debe notarse que el validador puede ser una parte íntegra del ambiente de alta-seguridad, pero también podría ser un componente separado que corre en el ambiente de alta-seguridad.

[0047] La FIG. 5 muestra un ejemplo de un validador 550 que realiza la aprobación de acuerdo con el procedimiento descrito anteriormente. El validador 550 tiene la habilidad de evaluar un cargador 410 para la exactitud y/o complacencia con algún conjunto de normas conocidas. Adicionalmente, el validador 550 tiene la habilidad para evaluar y/o afectar el estado de la máquina 502. Usando la combinación de estas habilidades, el validador 550 puede asegurar que el cargador 410 se comportará correctamente, asegurando que el cargador 410 que se espera que esté corriendo está, y asegurando que la máquina en que está corriendo está en un estado en que el cargador 410 se conoce para comportarse correctamente.

[0048] Estructuralmente, el validador 550 comprende dos componentes: una parte general 504, y una parte cargador-específica 506. La parte general "504" comprende código que es común a una variedad amplia de validadores (o todo el validador). La parte 506 cargador-específica es código que específicamente se relaciona a la aprobación de un cargador 410 particular - es

decir, el código que entiende que es una instancia correcta del cargador 410 (o estado 2 del cargador 410) parecería como si realizara pruebas para asegurar que el cargador 410 conforma esta comprensión. Así, la parte 506 cargador-específica puede combinarse con la parte 504 general para formar un validador completo.

[0049] El validador 550 es el componente que finalmente decide si la clave 204 se proporcionará al cargador (qué lo proporcionará entonces después al sistema operativo, para ser usado por el sistema operativo de acuerdo con la manera permitida por un programa de autenticación, como el descrito anteriormente). El validador 550 puede proteger la clave 204 sellando la clave 204 al validador 550 en una facilidad de almacenamiento sellado 508. La facilidad de almacenamiento sellado 508 puede ser un rasgo de alta-seguridad que calcula el ambiente. La facilidad de almacenamiento sellado 508 puede permitir a los componentes que corren en un ambiente de alta-seguridad para sellar los datos arbitrarios, tal que ningún otro componente puede recuperar los datos. Por ejemplo, la parte 506 cargador-específica del validador 550 puede sellar la clave 204 a sí mismo. La facilidad de almacenamiento sellado 508 impedirá a cualquier componente de otra manera que la parte 506 cargador-específica abra la clave 204. Así, la clave 204 es protegida para que sólo pueda obtenerse de la parte 506 cargador-específica, y la parte 506 cargador-específica distribuirá sólo la clave 204 a un cargador de la parte 506 cargador-específica que no causará que la clave 204 sea distribuida impropriamente. En otra personalización, la clave se sella basada en una combinación de componentes que incluyen el validador y la parte 2 de cargador 410. En tal personalización, la parte cargador-específica del validador 506 puede eliminarse, porque el propio mecanismo del almacenamiento sellado asegura la presencia de una fase correcta esencialmente cargador 2 que antes de que la clave pueda abrirse, desde el binario para una fase correcta conocida 2 el cargador es la parte de la métrica que va dentro del sellamiento.

#### **Proceso de ejemplo de Protección del SYSKEY**

**[0050]** La Fig. 6 muestra un proceso del ejemplo de proteger una clave de acuerdo con los aspectos de la invención descritos anteriormente. En algún punto después de que una máquina se inicia, un cargador del sistema operativo (602) se ejecuta. Durante la ejecución del cargador del sistema operativo, se validan el cargador y el estado de la máquina (604), asegurando por eso que el cargador se comportará de una manera predecible por las razones descritas anteriormente. Después de que el cargador y el estado de la máquina se han validado, el cargador se usa para cargar el sistema operativo mientras impide a los componentes falsos ser cargados (606). (Como discutido anteriormente, la validación del cargador y del estado de la máquina significa que el comportamiento futuro del cargador es conocida no para producir la carga de componentes falsos.) Después de que el sistema operativo está cargado, el programa de autenticación se ejecuta, el cual conduce el acceso a la clave.

#### **Otras Personalizaciones de Ejemplo**

**[0051]** Debe notarse que no pueden usarse los mecanismos de la invención presente meramente para realizar una carga del sistema operativo, pero generalmente puede usarse para permitir que un sistema realice alguna función en un estado raro (inválido), mientras requiere de la aprobación para proceder con ciertas funciones. Por ejemplo, una computadora puede realizar algunas funciones que pueden realizarse sin ninguna clase de aprobación (por ejemplo, la computadora puede actuar como una radio), pero la aprobación puede tener que ser realizada antes de que la computadora pueda realizar las funciones más sensibles (por ejemplo, leer los archivos del disco duro). En general, una máquina puede verse como un aparato y un PC general, y la parte del aparato no requiere la validación/autenticación (validation/logon), pero la parte general del PC la hace.

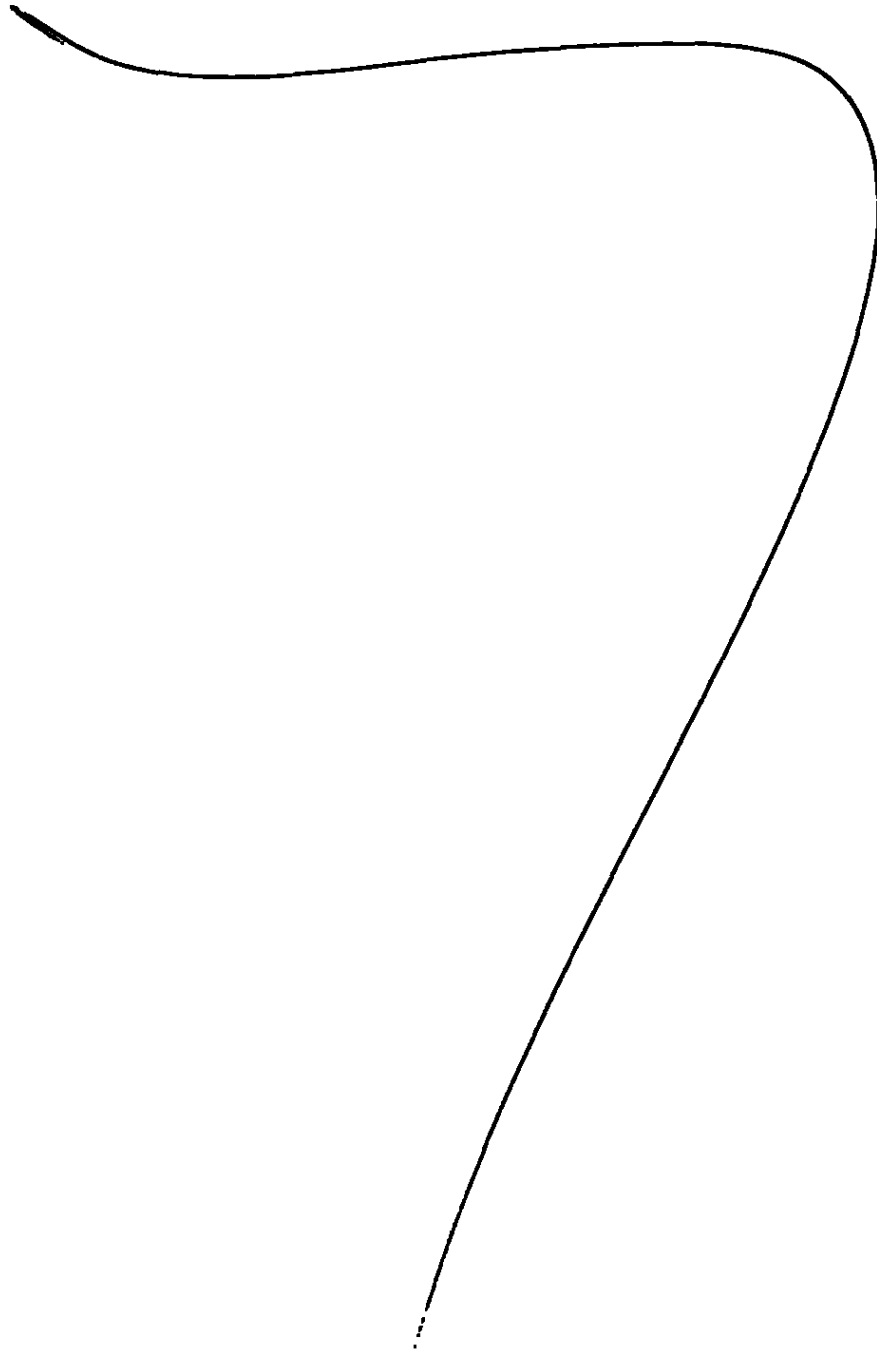
**[0052]** Es más, el validador no se limita a cualquier liberación/no liberación (releasing/not) del conjunto de claves del programa de autenticación, pero generalmente podría ser configurado para liberar conjuntos específicos de claves para pilas de software particulares - por ejemplo, hay una pila de software que puede conseguir una primera clave, pero el validador sólo podría liberar una segunda clave a una pila del "software difícilmente validada."

**[0053]** Adicionalmente, las claves necesariamente no necesitan ser liberadas a través de un procedimiento de autenticación, sino que puede proporcionarse a través de los procedimientos de aprobación arbitrarios. Por ejemplo, un jugador de DVD binario podría conseguir, por decir, las claves de función del DVD después de que el validador ha determinado que el jugador es el binario correcto, pero sin requerir una autenticación.

**[0054]** Adicionalmente, algunos programas podrían trabajar en alguna forma sin la aprobación, pero entonces requiere un tipo de aprobación antes de que le permita exponer cierta funcionalidad. Por ejemplo, la aplicación de la telefonía podría empezar sin autenticación requerida, pero puede requerir una autenticación antes de que las claves puedan distribuirse que permitiría a las funciones criptográficas de la aplicación trabajar.

**[0055]** Es notado que los ejemplos anteriores se han proporcionado meramente con el propósito de explicación y no han estado de ninguna manera para ser traducidos como limitantes de la invención presente. Mientras la invención se ha descrito con referencia a varias personalizaciones, se entiende que las palabras que se han usado aquí dentro son palabras de descripción e ilustración, en lugar de palabras de limitaciones. Más allá, aunque la invención se ha descrito aquí dentro con referencia a medios particulares, materiales y personalizaciones, no se piensa que la invención es limitada a los detalles descubiertos aquí dentro; más bien, la invención se extiende a todas las

estructuras funcionalmente equivalentes, métodos y usos, como está dentro del alcance de las demandas adjuntas. Esos experimentados en el arte, teniendo el beneficio de enseñar esta especificación, pueden efectuar numerosas modificaciones y cambios de esto sin partir del alcance y espíritu de la invención en sus aspectos.



**Lo que se Reivndica:**

1. Un medio legible por computadora codificado con las instrucciones legibles por computadora para realizar un método comprendiendo:

iniciar un cargador del sistema operativo;

validar la identidad o exactitud de dicho cargador;

asegurar que una máquina en la que dicho cargador de sistema operativo corre en un estado conocido; y

si la identidad o exactitud de dicho cargador se valida, y si dicha máquina en la cual corre el sistema operativo está en un estado conocido, entonces,:

proporcionar una clave a dicho cargador; y

permitir que dicho cargador cargue un sistema operativo.

2. El medio legible por computadora de la demanda 1, en donde el método más allá comprende:

ejecutar un programa de autenticación que conduzca el acceso a la clave.

3. El medio legible por computadora de la demanda 2, en qué dicho programa de autenticación asegura que un usuario proporcione las credenciales como una condición a dicho programa de autenticación concediendo el acceso a la clave.

4. El medio legible por computadora de la demanda 1, en qué dicha clave es útil para descifrar una partición del sistema operativo cifrada.

5. El medio legible por computadora de la demanda 1, en qué dijo acto de validar la identidad o exactitud de dicho cargador, y dicho acto de asegurar que una máquina está en un estado conocido, se realiza después de que todo lo siguiente se ha ejecutado:

un sistema básico de entrada salida;

una opción ROM;

un registro de arranque principal; y  
un sector de arranque.

6. El medio legible por computadora de la demanda 5, en qué dicho acto de validar la identidad o exactitud de dicho cargador, y dicho acto de asegurar que una máquina está en un estado conocido, se realiza después de que una parte del cargador del sistema operativo dicho se ha ejecutado.

7. El medio legible por computadora de la demanda 6, en qué dicho acto de validar se realiza después de que todo o substancialmente todo un sistema operativo ha empezado, o después de que algún número de particiones ha empezado, cada una de las particiones que comprenden un ambiente de algún grado de aislamiento de otras particiones es mantenida por un supervisor.

8. El medio legible por computadora de la demanda 1, en qué dicha clave es sellada para un validador que realiza dicha validación asegurando el acto y dicho acto de aseguramiento, en donde el método más allá comprende:

dicho validador abriendo dicha clave.

9. El medio legible por computadora de la demanda 8, en qué dicha clave es sellada por lo menos a dicho validador, y en qué dicho validador valida una parte de por lo menos dicho cargador.

10. El medio legible por computadora de la demanda 8, en qué dicha clave es sellada al validador dicho y a por lo menos una parte de dicho cargador.

11. El medio legible por computadora de la demanda 1, en qué dicha máquina comprende una máquina física.

12. El medio legible por computadora de la demanda 1, en qué dicha máquina comprende una máquina virtual.

13. El medio legible por computadora de la demanda 12, en qué dicha clave sólo se revela si una arquitectura de máquina virtual no ha cambiado o ha sido válida.

14. El medio legible por computadora de la demanda 1, en qué dicho acto de asegurar que una máquina está en un estado conocido comprende:

evaluar un estado actual de máquina y comparar dicho estado actual con el estado conocido dicho.

15. El medio legible por computadora de la demanda 1, en qué dicho acto de asegurar que una máquina está en un estado conocido comprende:

configurar un estado actual de máquina para ser consistente con dicho estado conocido.

16. El medio legible por computadora de la demanda 1, en qué dicho cargador es un sistema operativo completo, o una instancia completa de un sistema operativo.

17. Un sistema para realizar una arranque de un sistema operativo bajo circunstancias que proporcionan las convicciones acerca de la fiabilidad del arranque, el sistema comprende:

un validador que evalúa la exactitud o identidad de un cargador del sistema operativo que cargará el sistema operativo, y ese además evalúa un estado de una máquina en que dicho cargador del sistema operativo operará, el cual permite o no al cargador del sistema operativo para proceder con cargar el sistema operativo que depende de si o no la exactitud o identidad del cargador del sistema operativo se verifica, y qué lugares de dicha máquina estuvieron en un estado anterior conocido para permitir que dicho cargador del sistema operativo proceda.

18. El sistema de la demanda 17, en donde una clave es sellada a dicho validador, y en donde dicho validador abre dicha clave y proporciona dicho clave al cargador del sistema operativo dicho si dicho cargador del sistema operativo es permitido para proceder.

19. El sistema de la demanda 17, en qué dicho validador evalúa la exactitud o identidad del cargador del sistema operativo, y evalúa el estado de la máquina, después de que el cargador del sistema operativo ha empezado la operación, pero antes de que el cargador del sistema operativo haya inicializado un kernel o un controlador del dispositivo.

20. El sistema de la demanda 17, en qué dicho validador comprende:

una parte general que es común a una clase de validador; y

una parte específica que es específico al cargador del sistema operativo dicho, y eso es reemplazable con una parte diferente en el caso donde dicho validador está usándose para validar a un cargador del sistema operativo diferente.

21. El sistema de la demanda 17, en qué dicha máquina comprende una máquina física.

22. El sistema de la demanda 17, en qué dicha máquina comprende una máquina virtual.

23. Un método de arranque de un sistema operativo comprende:

ejecutar una entrada salida básica del sistema, una opción ROM, un registro de arranque principal, y un sector de arranque;

empezar a un cargador del sistema operativo;

validar dicho cargador del sistema operativo;

validar un estado de una máquina en que dicho cargador del sistema operativo se ejecuta;

si dicho cargador del sistema operativo, y dicho estado de máquina, se determina para ser válido, permitiendo entonces que dicho cargador del sistema operativo para cargue un sistema operativo.

24. El método de la demanda 23, en donde una clave que es necesaria realizar correctamente por lo menos una función bajo el sistema operativo dicho se sella a un validador que realiza dichos actos de validar a un cargador del sistema operativo y validar un estado de la máquina, y en donde el método más allá comprende:

si el cargador del sistema operativo y la máquina declaran que es válido, abriendo entonces dicha clave y proporcionando dicha clave al cargador del sistema operativo dicho.

25. El método de demanda 23, más allá, comprende:

después de que dicho sistema operativo se ha cargado, ejecutando un programa de autenticación que conduce el acceso a dicha clave.

26. El método de la demanda 23, en qué o dicho programa de autenticación permite o no que componentes de dicho sistema operativo usen dicha clave que depende de si un usuario completa un procedimiento de autenticación con éxito. . -

27. El método de demanda 23, en qué dichos actos de validar dicho cargador del sistema operativo y validando dicho estado de la máquina se realiza después de que el cargador del sistema operativo dicho ha realizado por lo menos una acción.

28. El método de la demanda 27, en qué dichos actos de validar dicho cargador del sistema operativo y validando el estado de dicha máquina se realiza a tiempo antes de restablecer el estado de la máquina que le impedirá al cargador funcionar para cargar correctamente dicho el sistema operativo.

29. El método de la demanda 23, en qué dicha máquina comprende una máquina física.

30. El método de la demanda 23, en qué dicha máquina comprende una máquina virtual.

### **RESUMEN DEL DESCUBRIMIENTO**

Un mecanismo para arranque del sistema operativo protegido que impide a los componentes falsos ser cargados con el sistema operativo, y además prevenir la divulgación de la clave del sistema bajo circunstancias impropias. Después de que una parte del procedimiento de inicio (startup) de máquina ha ocurrido, el cargador del sistema operativo se ejecuta, el cargador se valida, y un estado de la máquina correcto o se verifica para existir y/o se crea. Una vez el cargador se ha verificado que es un cargador legítimo, y el estado de la máquina bajo el cual está corriendo se verifica que es correcto, el comportamiento del futuro cargador se conoce para proteger contra la carga de componentes falsos que podrían causar divulgación de la clave del sistema. Con el comportamiento del cargador siendo conocido que es seguro para la clave del sistema, el validador puede abrir la clave del sistema y puede proporcionarlo al cargador.

## Ambiente de C3mputo 100

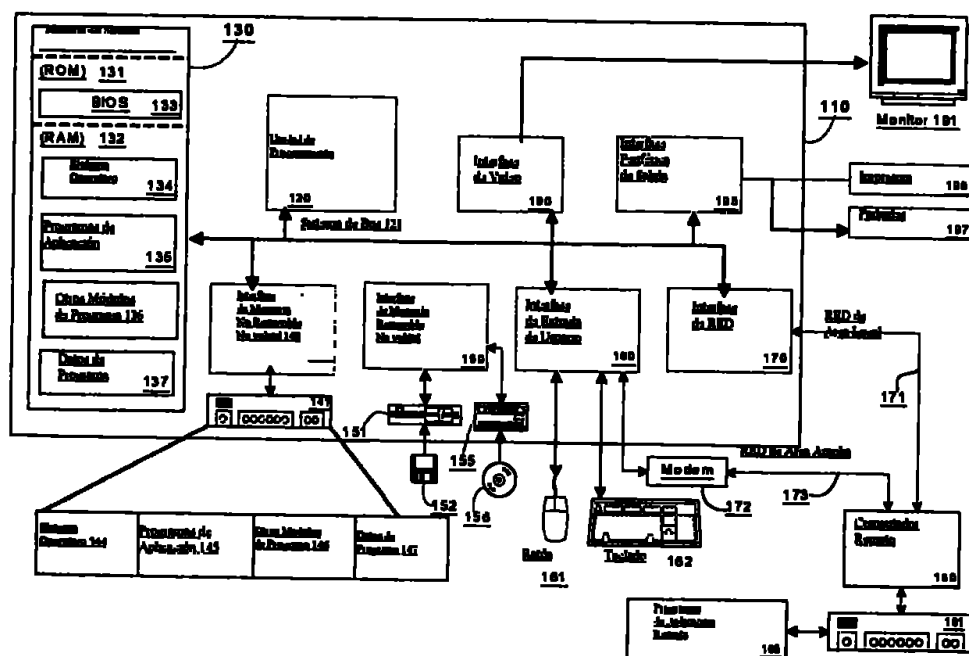
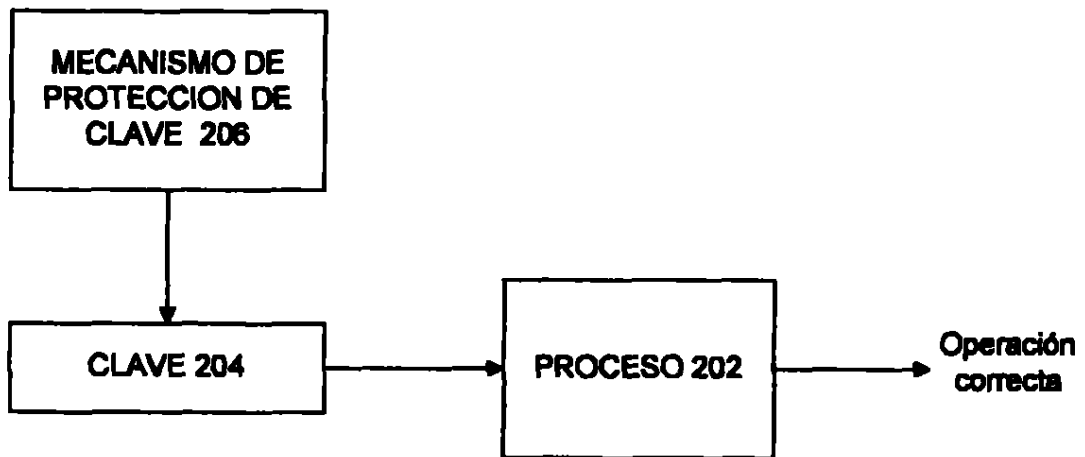
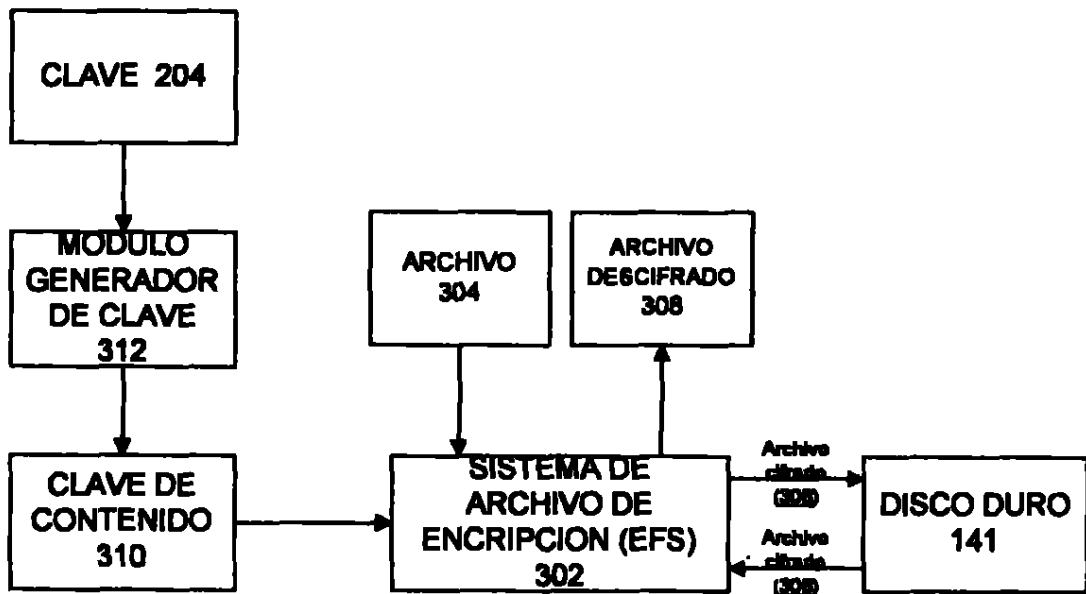


**FIG. 1**

2/5

**FIG. 2****FIG. 3**

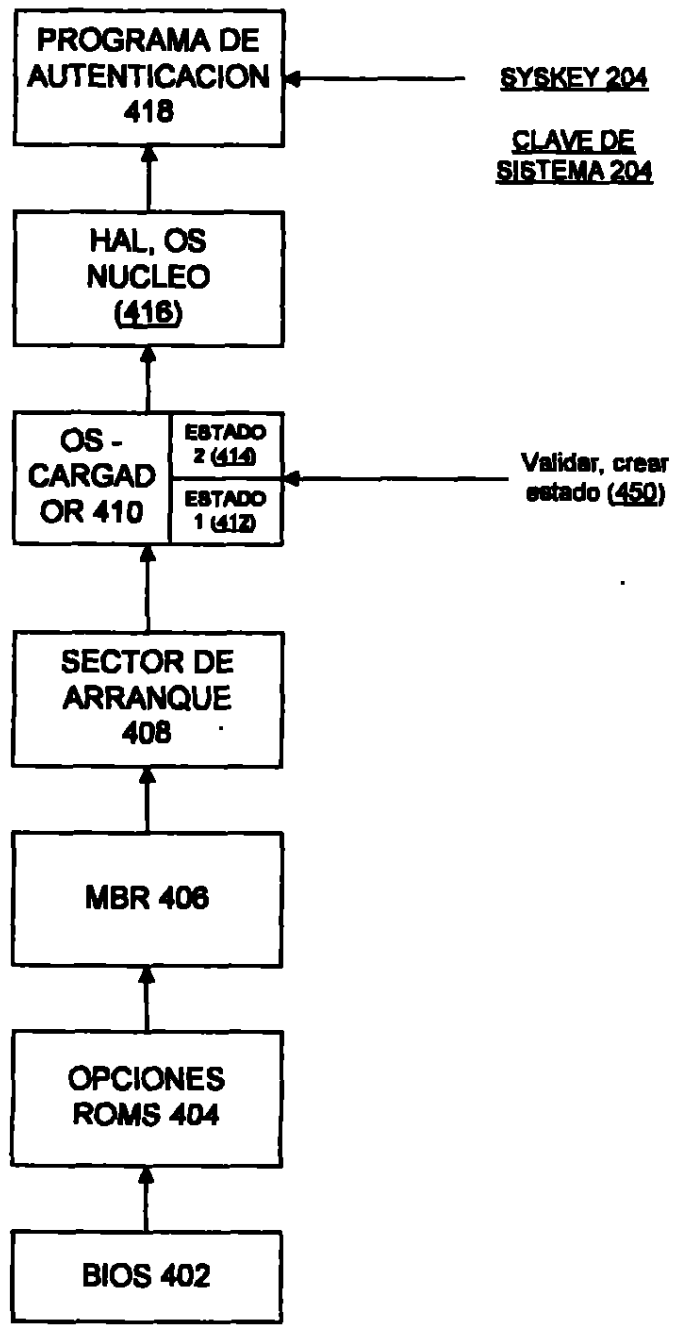


**FIG. 4**

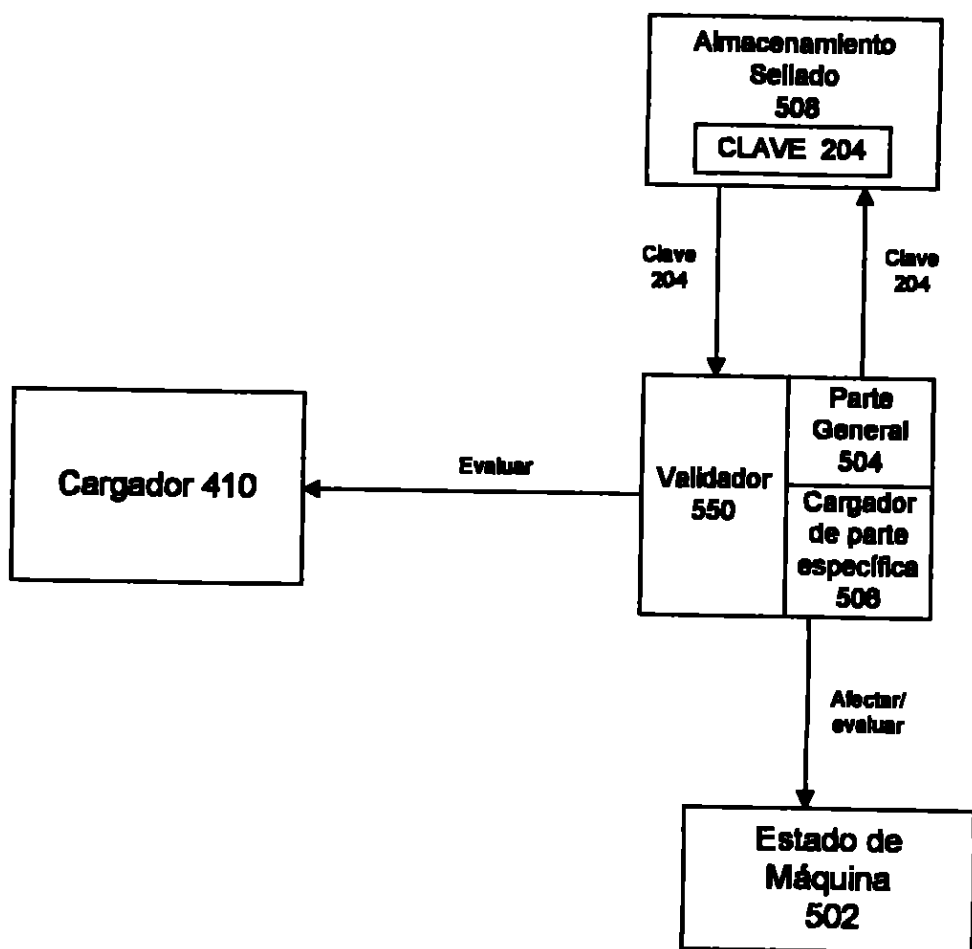
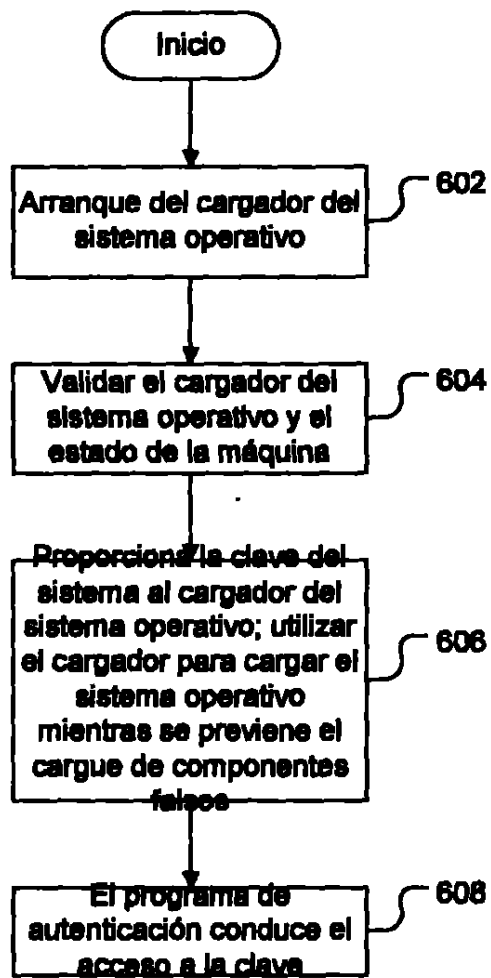


**FIG. 5**

5/5

**FIG. 6**

## **OSYSTEM AND METHOD FOR PROTECTED OPERATING SYSTEM BOOT USING STATE VALIDATION**

### **FIELD OF THE INVENTION**

[0001] The present invention relates generally to the field of computing. More particularly, the invention provides a mechanism to ensure that a system proceeds from a known safe state, and this mechanism can be used to boot a system in a way that provides sufficient assurance of the system's correct behavior. This assurance of correct behavior, in turn, can prevent one or more keys from being distributed under inappropriate circumstances.

### **BACKGROUND OF THE INVENTION**

[0002] Computer security is often dependent on being able to predict the behavior of software components. In general, the security of a system may flow from the premise that a known program whose behavior is understood, which proceeds from a known good state, will act in a predictable manner. Conversely, the thwarting of security – which may involve getting a computer system to behave in ways that are outside the contemplation of its designer – can generally be realized by replacing or changing a known program, or running it in a state in which its behavior is not understood. Thus, one aspect of providing security for a computing environment includes verifying that a known program is being used, and that it is proceeding from a known good state.

[0003] One area where predictability of behavior is particularly important is in the loading of an operating system and its components. Although the operating system itself may be designed to provide some level of trustworthiness as to its behavior, the time before such an operating system has been loaded is a time when the system is particularly vulnerable to attack, since the infrastructure that protects the operating system from attacks may not have been established yet (or may be in the process of being established). Thus, ensuring that the operating system loads in a predictable way is important for protecting the operating system from certain classes of attacks.

[0004] One type of security breach that can flow from non-secure loading of an operating system relates to the protection of the key (or keys) that enable certain restricted functionality. By way of example but not limitation, the MICROSOFT WINDOWS operating systems employ a

---

system key, or "SYSKEY," that is used to protect various processes by making the correct performance of those processes dependent on the availability of SYSKEY. For example, the key needed to decrypt private information that is stored by the operating system in encrypted form may be derivable from the SYSKEY.

[0005] Conventionally, the keys needed to perform restricted operations are protected by the logon procedure. Typically, the user must correctly authenticate himself (e.g., by providing correct logon credentials, such as a username/password combination) prior to commencing use of the system. Use of the keys is enabled only if the user correctly authenticates, and the system will only permit the user a limited number of tries (e.g., three) before concluding that the user has failed to logon properly. (This type of limit on the number of attempts to logon prevents unauthorized users from enabling use of protected functionality by using a brute force attack to guess the password in the case of, say, a stolen laptop computer.) However, using the logon procedure to protect access to keys assumes that the operating system loader correctly loaded the operating system with the correct logon program, and that the use of the keys has not been otherwise enabled by rogue code that may be running. If a rogue loader was used instead, and the rogue loader causes a rogue logon program to be loaded with the operating system, then the use of keys might be enabled, or the keys might even be divulged, without the correct credentials having been entered. Since the loading of the operating system provides an opportunity for a security breach, protection of the keys in such a situation requires that the loading of the operating system take place under circumstances where it can be verified to take place correctly.

[0006] One problem that occurs with verifying the security of an operating system load process is that legitimate operating system loads can involve many different programs (e.g., there are numerous different "option ROMs," which are pre-OS programs that run during a system boot procedure), and there are numerous different procedures that can be performed as part of an operating system load. Thus, there are a nearly uncountable number of different legitimate machine states during a load, and identifying all such states and verifying that the machine is in a known good state may prove to be an infeasible task. However, not all parts of the load procedure have security implications. It may be more efficient to let the load proceed without any attempt to evaluate its security, but then set the environment to a known good state before starting any procedure that could affect a security-related function, such as the distribution of keys. More

generally, an arbitrary system can be allowed to run for some time without any type of security evaluation, as long as the system can be set to a known good state before allowing any actions that have security implications to take place.

[0007] In view of the foregoing, there is a need for a mechanism that overcomes the drawbacks of the prior art.

## **SUMMARY OF THE INVENTION**

[0008] The present invention provides for the loading of an operating system under circumstances where the load can be verified to be performed correctly. When a machine is started, the very early startup procedure (e.g., BIOS, option ROMs, master boot record, boot sector, etc.) are performed. After these early procedures are performed, the operating system loader is started, and may perform various preliminary tasks. After the operating system loader is started and has performed such preliminary tasks, a validation of the operating system loader is performed.

[0009] The validation comprises performing validity tests on the loader itself, or a portion of the loader (e.g., checksums, or other tests that are aimed at evaluating the identity and correctness of the loader), as well as evaluating the current machine state (or forcing the machine to conform to a known good state). If the loader (or relevant portion) is known to be correct, and the machine state is one in which the loader has previously been verified to behave correctly, then the behavior of the loader can be predicted. Thus, it can be assured that a correct loader operating under a correct machine state will not load a component that would cause data that enables protected functions (e.g., cryptographic keys, such as the system key) to be distributed under inappropriate circumstances.

[0010] Preferably, the validation is performed by a validator that runs in a high-assurance environment. A high assurance environment is one in which it is possible to provide a relatively high degree of assurance that processes performed in it will run in the manner they are expected to. Thus, the assurance that the validator will operate correctly is derived from the fact that the validator is verified by a process that runs in the high-assurance environment, (e.g., by verifying a signature of its binary), and from the underlying trust that processes in the high-assurance environment will be carried out correctly. (Or, at least, there is a level of assurance that the high-assurance environment will not interfere with, or permit interference with, the correct operation of a

process that operates within such an environment; one must still have a separate basis of trust that the program that implements a process within the high-assurance environment will correctly perform in the manner that it is expected to perform.) A high-assurance environment may provide sealed storage, which is a storage facility in which data can be sealed to a particular object (e.g., to a particular program), and that the high-assurance environment control in such a way that the sealed data will not be released to any object other than the one to which the data is sealed (as verified by the high-assurance environment). The validator can use this sealed storage to seal a key (e.g., the SYSKEY) to itself, and can refuse to unseal the key except to the correct object, and when the circumstances (e.g., the machine state) are satisfactory under some standard.

[0011] Other features of the invention are described below.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

[0012] The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings example constructions of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

[0013] FIG. 1 is a block diagram of an example computing environment in which aspects of the invention may be implemented;

[0014] FIG. 2 is a block diagram of a system that employs a process whose correct operation is dependent on a system key;

[0015] FIG. 3 is a block diagram of an encrypted file system that protects encrypted data from unauthorized decryption by making decryption dependent on a system key;

[0016] FIG. 4 is a flow diagram of an example boot process with validation in accordance with aspects of the invention;

[0017] FIG. 5 is a block diagram of an example validator in accordance with aspects of the invention; and

[0018] FIG. 6 is a flow diagram of an example process of protecting a system key in accordance with aspects of the invention.

## **DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS**

### **Overview**

[0019] Several processes that may take place under an operating system are dependent on one or more keys for their correct operation. Access to the keys may be controlled by an authentication program, such as a logon program, which refuses to enable use of the key(s) unless the user provides correct credentials, such as a username/password combination. Thus, by the logon program's refusal to enable the use of keys in the absence of correct credentials, several processes (e.g., decryption of encrypted files) can be disrupted (or entirely prevented) for users who do not know the password. While the logon program may be effective at gating access to the keys, an operating system loader could be tricked into loading a different component that would distribute the keys outside of the authentication rules imposed by the logon program. Thus, when keys are distributed in this manner, protecting the keys requires protecting the operating system loading process. The present invention provides mechanisms that can be used to protect the loading process.

### **Exemplary Computing Arrangement**

[0020] FIG. 1 shows an exemplary computing environment in which aspects of the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0021] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, embedded systems, distributed computing environments that include any of the above systems or devices, and the like.

[0022] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules

include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules and other data may be located in both local and remote computer storage media including memory storage devices.

[0023] With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The processing unit 120 may represent multiple logical processing units such as those supported on a multi-threaded processor. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus). The system bus 121 may also be implemented as a point-to-point connection, switching fabric, or the like, among the communicating devices.

[0024] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer

51

110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0025] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0026] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0027] The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program

modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0028] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0029] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the

computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

#### Action protected by keys

[0030] A computing environment may employ a key on which certain processes that take place within the environment depend for their correct operation. The system key, or "SYSKEY" used by the MICROSOFT WINDOWS operating systems is one example of such a key, but is not a limiting example. In a preferred embodiment, a key on which a process depends is a unique, cryptographically random per-platform key – i.e., given two machines, it is likely that the two machines have different keys. Thus, a process that is dependent on such keys is unlikely to be portable from one platform to another – at least to the extent that effective mechanisms are employed to ensure that a platform's key is unavailable outside of that platform.

[0031] FIG. 2 shows an example system in which a process runs that is dependent on a key. Process 202 is dependent on key 204 in order to operate correctly. It should be noted that process 202 is not limited to the traditional notion of a process – i.e., a unit of execution that can be managed by an operating system and assigned an address space – but rather refers more generally to any operation or series of operations that can be performed on a computer. It should be further noted that, while this example shows a process that is dependent on a cryptographic key, the term "process" as used herein is not limited to processes that perform cryptographic operations.

[0032] As shown in FIG. 2, if key 204 is available as an input to process 202, then process 202 operates correctly. On the other hand, if key 204 is not available as an input to process 202, then process 202 does not operate correctly. A key protection mechanism 206 controls access to key 204 – i.e., mechanism 206 either provides, or does not provide, process 202 with key 204, depending on whether the relevant security conditions have been met. For example, a user may have to log in and provide a correct password before mechanism 206 will enable use of key 204.

[0033] It should be noted that the denial of key 204 in order to prevent process 202 from operating correctly is sometimes a desired result. For example, the encryption/decryption of files is one example of a process 202 that is protected by key 204. The correct decryption of a file may be

dependent on access to key 204. If the user cannot correctly log in and authenticate himself, then it may be desirable for the decryption of a file not to proceed, since the inability of a user to log in may indicate that the computer is being operated by someone other than its intended user (e.g., in the case of a stolen laptop). Thus, key protection mechanism 206 can make access to key 204 dependent on the relevant security conditions being met, and can use denial of key 204 to disrupt processes that need to be disrupted when those security conditions are not met. Mechanism 206's ability to disrupt processes in this manner is dependent on those processes' requiring key 204 for their correct operation, since it is this dependency that enables the act of denying key 204 to disrupt the process.

[0034] FIG. 3 shows a specific (but non-limiting) example of a process that is dependent on key 204. In the example of FIG. 3, the example process is an Encrypting File System (EFS) 302 that stores files in encrypted form, and also decrypts the encrypted files. It should be noted that one purpose of such file encryption is to protect data on a laptop computer from being recovered by a thief in the event that the laptop is stolen. When a file 304 is generated for storage, the file is provided to EFS 302. EFS 302 then encrypts file 304, and turns file 304 into encrypted file 306, which is stored on hard disk 141. When a request is made to retrieve encrypted file 306, EFS 302 retrieves encrypted file 306 and decrypts it to generate decrypted file 308. (In practice, the substance of decrypted file 308 is the same as that of original file 304, although for clarity FIG. 3 shows these two instances of the file separately: file 304 is the original file, and decrypted file 308 is the same file after it has been encrypted, stored, retrieved, and decrypted by EFS 302.)

[0035] It will be noted that one input to EFS 302 is content key 310. Content key 310 is preferably a symmetric key that functions as an input to a cryptographic method. Content key 310 is used to encrypt file 304 in order to create encrypted file 306, and is also used to decrypt encrypted file 306 in order to create decrypted file 308. It can be appreciated that storing key 310 in some easily retrievable place could quickly render EFS 302's ability to protect the data ineffective: If the content key is easily available on a hard disk, or if it is easily derivable from some known feature of the laptop (e.g., the processor serial number), then it would make no difference whether the files were stored in encrypted form, since a thief could easily find the key and decrypt them. Thus, it is desirable to protect the key by deriving it in some way that can only be done with the cooperation of the true owner. One way to protect the key is to use a key generator module 312, which receives key

55

204 as input, and derives content key 310 as a function of key 204. Thus, to the extent that key 204 will only be provided under appropriately secure circumstances, it is likewise true that content key 310 will only be derivable under appropriate circumstances. In other words, by making the derivation of content key 310 dependent on the availability of key 204, whatever protection is afforded to key 204 can be extended to content key 310. For example, if providing key 204 requires that the user log in by providing a correct password, then it can be assured that content key 310 will not be available unless the user correctly logs in.

[0036] Thus, protecting key 204 from being given out under the wrong circumstances is important, since other processes may depend on key 204 being given out only when the correct security context (e.g., a legitimate, logged in user, who tenders the correct password) is present. As described below, one way to cause a machine to distribute key 204 in a manner that may result in misuse is to boot the machine in an insecure environment in which rogue components of the operating system can replace the correct components (where the correct components are presumed to be able to protect key 204). Thus, mechanisms are described below that ensure that a machine is booted in a known (safe) environment before key 204 can be distributed.

#### Boot Process with State Validation

[0037] FIG. 4 shows a typical sequence of events that are used in a boot procedure.

[0038] Initially, the machine is powered on. A typical machine is configured to begin executing instructions at some fixed address at the time that it is powered on. The instructions typically contained at this address are known as a "BIOS" 402, or "Basic Input/Output System." At the conclusion of BIOS 402's execution, BIOS 402 starts execution of small programs called "option ROMs" 404. Option ROMs are programs that perform very basic early-boot functions, such as setting the hardware password for the machine, or selecting which of several operating systems is to be booted. After option ROMs 404 execute, the machine is instructed to load the Master Boot Record (MBR) 406.

[0039] MBR 406 is an executable program. Typically, MBR 406 resides on the first sector of a computer's hard disk, and begins by looking up which partition to use for further booting in the partition table. (For example, a disk may be partitioned for use with different operating systems, and each operating system may require a different boot procedure.) After the correct partition has been

looked up, MBR 406 transfers control to the boot sector 408 associated with that partition. Boot sector then commences the process of loading the operating system loader 410 that will ultimately load the operating system. It should be noted that MBR 406 is shown in FIG. 4 merely to show how such a component would fit into an example boot process, and the invention is not limited to boot procedures that use MBR 406.

[0040] During the time that operating system loader 410 is running, the loader and machine state are validated (450). ("Machine," in this context, can refer to either a physical machine or a virtual machine.) The validation is performed by a trusted software component that operates in a high-assurance environment (an embodiment of which is more particularly described below), and therefore there is some degree of assurance/trustworthiness that the validation of loader 410 is being done correctly. In essence, by validating the relevant aspects of the machine state while operating system loader 410 has control of the machine, and by validating loader 410, it is possible to provide some level of assurance that illegitimate or rogue versions of operating system components (e.g., the hardware abstraction layer, the kernel, drivers, etc. (416)) will not be loaded later on, up to the time that logon program 418 is run. Preventing rogue components from being loaded up to the time that logon program 418 is run is important because one of the first things that the operating system will do after it is loaded is run a logon program 418 that gates access to key 204, and if rogue components could be loaded, then these components could cause logon program 418 to misbehave in a manner that could result in distributing key 204 under inappropriate circumstances, thereby compromising the security of all components that depend on the protection of key 204 as described above. Thus, protection of key 204 can be achieved by exerting tight control over the state of the machine from the time that the operating system loader 410 is run until the logon program has completed.

[0041] Validation 450 takes place during the time that operating system loader 410 is running. Validation includes validating the loader and the machine state, and may also include setting the machine state to a known good state. The basic idea behind validating and setting the machine state is to place the machine in a state where, if the loader executes while the machine is in that state, then the loader will not load any rogue components or otherwise behave in ways that would result in security breaches. Validation 450 ensures that the loader code is, in fact, the code that has previously been verified to behave correctly, and also ensures that the machine is in a state

under which this known code will behave correctly (either by verifying that the machine already is in that state, or by placing the machine in that state). It will be appreciated that this technique is able to leverage the existing (safe) state of the machine to ensure that the future states of the machine will also be safe, and this leveraging is made possible by the fact that the behavior of the operating system loader 410 is not only known and understood, but also tightly circumscribed. Based on this observation, it will be appreciated that validation 450 should not take place too early (e.g., at the time that option ROMs 404, or MBR 406, is executed), since the wide variety of code from numerous different sources, and the vast breadth of states that the machine could be placed in as a result of executing such code, makes it difficult, if not impossible, to determine the behavior of the machine while executing all of these various different code modules. Thus, it is preferable not to be concerned with what states the machine is in prior to the running of loader 410, as long as the machine can be placed in a legitimate state at the time that loader 410 is running.

[0042] In a preferred embodiment, the execution of loader 410 is divided into two stages: stage 1 (412) and stage 2 (414). Preferably, before stage 2 is entered, the code that implements stage 2 is validated (450), and the trusted code that performs validation 450 then jumps into stage 2 at a well-defined entry point. (The "trusted code" that performs validation is the above-mentioned program that runs in a high-assurance environment. It will be understood by those of skill in the art that "trusted" does not imply absolute infallibility, but merely means that there is some basis to assume that the code will do its job correctly. Since the behavior of a program can be affected by the environment in which it runs, executing the trusted code in a high-assurance environment means that the trusted code will operate correctly: (1) the trusted code can be trusted to perform its function correctly in an environment that conforms to some expectations, and (2) the high-assurance environment can be trusted to correctly provide an environment that conforms to such expectations.) Stage 2 may then validate (according to some defined standard) any information that it has received from stage 1. The dividing line between stage 1 and stage 2 reflects the fact that there may be some aspects of a boot procedure that can be performed without any security verification, and without having any consequences for some security-related task (such as the distribution of key 204), as long as the relevant programs and state can be validated at some point before events could be set in motion that could cause such a security-related task not to be performed correctly. Moreover, there may be problems with attempting to validate state too early in the boot process: the actual execution

path that a legitimate boot procedure can take is quite variable, so it becomes difficult to define the difference between valid or invalid machine states under such variable circumstances. It may make sense to simply allow the load to proceed through any of these variable states, without trying to determine whether any state that the machine passes through is valid. Stage 1 thus represents the portion of the load that proceeds without any validation. At the end of stage 1, the loader and machine state are validated, and stage 2 begins. The presumption of such a dividing line is that processes can execute without any security restrictions, up to a certain point – at which point all relevant factors about the machine are validated and the machine is placed in a known good state, which essentially nullifies the effect of any prior actions that might have placed the machine in a state that is unacceptable from a security standpoint. The exact point in the load process where the dividing line between stage 1 and stage 2 is drawn is highly specific to the circumstances (e.g., What does the loader code look like? What tasks need to be done as part of the load?), and represents a sort of tradeoff: On the one hand, the dividing line should be late enough that, for future actions, the amount of variation in legitimate load behavior is small enough that legitimate behavior can feasibly be distinguished from illegitimate behavior. (As noted above, in the early load procedure, the large number of option ROMs and other variables make the number of possible execution paths so great that it is difficult to distinguish legitimate behaviors from illegitimate behaviors.) On the other hand, the dividing line should be early enough that it precedes any events (e.g., the loading of the logon program) that could have an affect on security, such as causing keys to be distributed inappropriately. In general, the dividing line between stage 1 and stage 2 allows one to run an arbitrary system in a “raw” (or “open,” or “unvalidated”) state for some time, and then start whatever components are necessary to verify behavior (e.g., a high-assurance environment), and then use such components to validate the current state of the machine (or force the machine into a known good state), at which point the process of doing something that can later affect security (e.g., the distribution of keys) can be allowed to proceed.

#### Validation 450

[0043] Validation 450 is, in essence, the act of verifying that the operating system loader 410 (or, more specifically, in a preferred embodiment, verifying that stage 2 of the loader) is a known trusted program, and ensuring that the machine on which it is running is in an known good

state. Thus, validation has two parts: (1) examining the loader program (or stage 2 of the loader program) to ensure that it is the known, trusted program that it is believed to be, and (2) changing the relevant machine state to a known good state under which the loader is known to behave correctly. The premise of performing (1) and (2) is that a known program, operating on a known machine state, will behave in a known way.

[0044] The first part of the validation – i.e., examining the loader – can be performed in a variety of ways. In the simplest example, there may be a hash of the known program (which may be cryptographically signed to create a digital signature, or else stored in some non-spoofable way), against which the actual program that is running can be compared. The problem with taking a hash over the entire program is that different instances of a legitimate program may have slightly different images, so a hash that is computed in a way that requires complete identity between the known program and the running instance of the program may be too restrictive. Preferably, validation is performed in a way that ensures that the program is what it is supposed to be, but that is not overly restrictive – e.g., the validation process may compare hashes of program components that are known to be fixed, and may perform other tests on program components that may change over time. The component that performs the validation should be tailored to the specifics of the program that is being validated; the invention is not limited to any particular technique for validating a program.

[0045] The second part of the validation – i.e., verifying/setting the machine state – is preferably performed by setting all of the relevant “sources” of state to known values. Typically, the relevant state that can affect an operating program comes from three sources: the CPU, the chipset, and the memory. Thus, these items can be put into a known good state at the time of validation – e.g., the CPU can be put into a known state – e.g., ring 0, with the program counter pointed at a known location, all data registers set to zero, and all memory other than the memory in which the loader program is stored being set to zero. If the loader has been verified to behave correctly when it runs with the machine in this state, then the combination of examining the loader and placing the machine in the known good state should ensure correct behavior up to the time that logon program 418 is run. As discussed above, one of the benefits (although not the only benefit) of correct behavior up to the running of logon program 418 is that rogue components will not be loaded that would cause key 204 to be distributed under inappropriate circumstances.

[0046] It should be noted that, in order for the validation process to protect the load of the operating system from rogue components, there should be sufficient assurance that the validation process itself is being performed correctly. Correctness of the validation process can be ensured by embodying the validator as a trusted agent that runs on or in a high-assurance computing environment. For example, there may be a small, operating system that performs a limited set of functions, but which provides a high degree of assurance that such an operating system will perform according to its specifications. Such an operating system could be run together with other operating systems on a single machine, and the isolation of such a high assurance operating system from the other (less secure) environments on the system could be enforced by a supervising component, such as a hypervisor or virtual machine monitor. (In one embodiment, a hypervisor enforces partitions, which are environments that the hypervisor maintains in some state of mutual isolation from each other, and in which an operating system can run.) Moreover, the high assurance component may have exclusive access to a root of trust, such as a hardware module that applies, and strongly protects, per-platform cryptographic keys. The validator could be a program (or "agent") that runs in such a high assurance environment, which would provide assurance that – to the extent that the high-assurance environment itself can be trusted to resist attacks– the validator would not be subject to attacks from outside the high-assurance environment that would cause it to behave incorrectly. Additionally, as noted above the high-assurance environment could provide sealed storage (i.e., the ability to store some datum and to release that datum only to a particular object to which the datum has been sealed), and the validator could use this sealed storage to store any keys that are to be distributed. It should be noted that the validator could be an integral part of the high-assurance environment, but could also be a separate component that runs in the high-assurance environment.

[0047] FIG. 5 shows an example of a validator 550 that performs validation in accordance with the procedure described above. Validator 550 has the ability to evaluate loader 410 for correctness and/or compliance with some set of known standards. Additionally, validator 550 has the ability to evaluate and/or affect machine state 502. Using the combination of these abilities, validator 550 can ensure that loader 410 will behave correctly, by ensuring that loader 410 is the loader that is expected to be running, and by ensuring that the machine on which it is running is in a state in which loader 410 is known to behave correctly.

**[0048]** Structurally, validator 550 may comprise two components: a general portion 504, and a loader-specific portion 506. "General portion" 504 comprises code that is common to a wide variety of validators (or all validators). Loader-specific portion 506 is code that specifically relates to the validation of a particular loader 410 – i.e., the code that understands what a correct instance of loader 410 (or stage 2 of loader 410) would look like, and that performs tests to ensure that loader 410 conforms to this understanding. Thus, loader-specific portion 506 can be combined with general portion 504 to form a complete validator.

**[0049]** Validator 550 is the component that ultimately decides whether key 204 will be provided to the loader (which will then later provide it to the operating system, for use by the operating system in accordance with the manner permitted by a logon program, as described above). Validator 550 may protect key 204 by sealing key 204 to validator 550 in a sealed storage facility 508. Sealed storage facility 508 may be a feature of a high-assurance computing environment. Sealed storage facility 508 may permit components running in a high-assurance environment to seal arbitrary data to themselves, such that no other component can retrieve the data. For example, the loader-specific portion 506 of validator 550 may seal key 204 to itself. Sealed storage facility 508 will prevent any component other than loader-specific portion 506 from unsealing the key 204. Thus, key 204 is protected because it can only be obtained from the loader-specific portion 506, and loader-specific portion 506 will only distribute key 204 to a loader that loader-specific portion 506 believes will not cause key 204 to be distributed inappropriately. In another embodiment, the key is sealed based on a combination of components that includes the validator and stage 2 of loader 410. In such an embodiment, the loader-specific portion of the validator 506 can be eliminated, because the sealed storage mechanism itself essentially ensures the presence of a correct stage 2 loader before the key can be unsealed, since the binary for a known correct stage 2 loader is part of the metric that goes into the seal.

#### Example Process of Protecting SYSKEY

**[0050]** FIG. 6 shows an example process of protecting a key in accordance with the aspects of the invention described above. At some point after a machine is started, an operating system loader is executed (602). During the execution of the operating system loader, the loader and machine state are validated (604), thereby ensuring that the loader will behave in a predictable

manner for the reasons described above. After the loader and machine state have been validated, the loader is used to load the operating system while preventing rogue components from being loaded (606). (As discussed above, validation of the loader and machine state means that the future behavior of the loader is known not to result in the loading of rogue components.) After the operating system is loaded, the logon program is run, which gates access to the key.

#### Other Example Embodiments

[0051] It should be noted that the mechanisms of the present invention can be used not merely to perform an operating system load, but can be used generally to allow a system to perform some function in a raw (unvalidated) state, while requiring validation to proceed with certain functions. For example, a computer may perform some functions that can be performed without any sort of validation (e.g., the computer can act as a radio), but validation may have to be performed before the computer can perform more sensitive functions (e.g., reading files from the hard disk). In general, a machine can be viewed as both an appliance and a general PC, and that the appliance part doesn't require validation/logon, but the general PC part does.

[0052] Moreover, the validator is not limited to either releasing/not releasing a particular set of keys to the logon program, but could more generally be configured to release specific sets of keys to particular software stacks – e.g., there is one software stack that can get a first key, but validator might release a second key only to a “more heavily validated” software stack.

[0053] Additionally, keys do not necessarily need to be released through a logon procedure, but rather can be provided through arbitrary validation procedures. For example, a DVD player binary could get, say, the DVD function keys after the validator has determined that the player is the correct binary, but without requiring a logon.

[0054] Additionally, some programs could work in some form without validation, but then require a type of validation before being permitted to expose certain functionality. For example, telephony application could start with no logon required, but may require a logon before the keys can be distributed that would permit cryptographic functions of the application to work.

[0055] It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the invention has been described with reference to various embodiments, it is understood that the words

which have been used herein are words of description and illustration, rather than words of limitations. Further, although the invention has been described herein with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed herein; rather, the invention extends to all functionally equivalent structures, methods and uses, such as are within the scope of the appended claims. Those skilled in the art, having the benefit of the teachings of this specification, may effect numerous modifications thereto and changes may be made without departing from the scope and spirit of the invention in its aspects.

**What is Claimed:**

1. A computer-readable medium encoded with computer executable instructions to perform a method comprising:

starting an operating system loader;

validating the identity or correctness of said loader;

ensuring that a machine on which said operating system loader runs is in a known state; and

if the identity or correctness of said loader validates, and if said machine on which said operating system runs is in a known state, then:

providing a key to said loader; and

allowing said loader to load an operating system.

2. The computer-readable medium of claim 1, wherein the method further comprises:

running a logon program that gates access to said key.

3. The computer-readable medium of claim 2, wherein said logon program ensures that a user provides credentials as a condition to said logon program granting access to said key.

4. The computer-readable medium of claim 1, wherein said key is usable to decrypt an encrypted operating system partition.

5. The computer-readable medium of claim 1, wherein said act of validating the identity or correctness of said loader, and said act of ensuring that a machine is in a known state, is performed after all of the following have executed:

a basic input output system;

an option ROM;

a master boot record; and

a boot sector.

6. The computer-readable medium of claim 5, wherein said act of validating the identity or correctness of said loader, and said act of ensuring that a machine is in a known state, are performed after a portion of said operating system loader has executed.

7. The computer-readable medium of claim 6, wherein said act of validating is performed after all or substantially all of an operating system has started, or after some number of partitions have started, each of said partitions comprising an environment from which some degree of isolation from other partitions is maintained by a hypervisor.

8. The computer-readable medium of claim 1, wherein said key is sealed to a validator that performs said validating act and said ensuring act, and wherein the method further comprises:  
said validator unsealing said key.

9. The computer-readable medium of claim 8, wherein said key is sealed to at least said validator, and wherein said validator validates at least a portion of said loader.

10. The computer-readable medium of claim 8, wherein said key is sealed to said validator and to at least a portion of said loader.

11. The computer-readable medium of claim 1, wherein said machine comprises a physical machine.

12. The computer-readable medium of claim 1, wherein said machine comprises a virtual machine.

13. The computer-readable medium of claim 12, wherein said key is revealed only if an architecture of said virtual machine has not changed or is valid.

14. The computer-readable medium of claim 1, wherein said act of ensuring that a machine is in a known state comprises:

evaluating a current state of said machine and comparing said current state with said known state.

15. The computer-readable medium of claim 1, wherein said act of ensuring that a machine is in a known state comprises:

setting a current state of said machine to be consistent with said known state.

16. The computer-readable medium of claim 1, wherein said loader is an entire operating system, or an entire instance of an operating system.

17. A system for performing a boot of an operating system under circumstances that provide assurances as to the reliability of the boot, the system comprising:

a validator that evaluates the correctness or identity of an operating system loader that will load the operating system, and that further evaluates a state of a machine on which said operating system loader will operate, which either allows or disallows the operating system loader to proceed with loading the operating system depending on whether or not the correctness or identity of the operating system loader is verified, and which places said machine in a known state prior to allowing said operating system loader to proceed.

18. The system of claim 17, wherein a key is sealed to said validator, and wherein said validator unseals said key and provides said key to said operating system loader if said operating system loader is allowed to proceed.

19. The system of claim 17, wherein said validator evaluates the correctness or identity of the operating system loader, and evaluates the state of the machine, after the operating system loader has started operation, but before the operating system loader has initialized a kernel or a device driver.

20. The system of claim 17, wherein said validator comprises:

a general portion that is common to a class of validators; and

a specific portion that is specific to said operating system loader, and that is

replaceable with a different portion in the case where said validator is being used to validate a different operating system loader.

21. The system of claim 17, wherein said machine comprises a physical machine.

22. The system of claim 17, wherein said machine comprises a virtual machine.

23. A method of booting an operating system comprising:

executing a basic input output system, an option ROM, a master boot record, and a boot sector;  
starting an operating system loader;  
validating said operating system loader;  
validating a state of a machine on which said operating system loader executes;  
if said operating system loader, and said state of said machine, are determined to be valid, then allowing said operating system loader to load an operating system.

24. The method of claim 23, wherein a key that is necessary to correctly perform at least one function under said operating system is sealed to a validator that performs said acts of validating an operating system loader and validating a machine state, and wherein the method further comprises:

if the operating system loader and the machine state are valid, then unsealing said key and providing said key to said operating system loader.

25. The method of claim 23, further comprising:

after said operating system has been loaded, executing a logon program that gates access to said key.

26. The method of claim 23, wherein said logon program either allows or disallows components of said operating system to use said key depending on whether a user successfully completes an authentication procedure.

27. The method of claim 23, wherein said acts of validating said operating system loader and validating said machine state are performed after said operating system loader has performed at least one action.

28. The method of claim 27, wherein said acts of validating said operating system loader and validating said machine state are performed at a point in time before resetting the machine state will prevent the loader from functioning correctly to load said operating system.

29. The method of claim 23, wherein said machine comprises a physical machine.

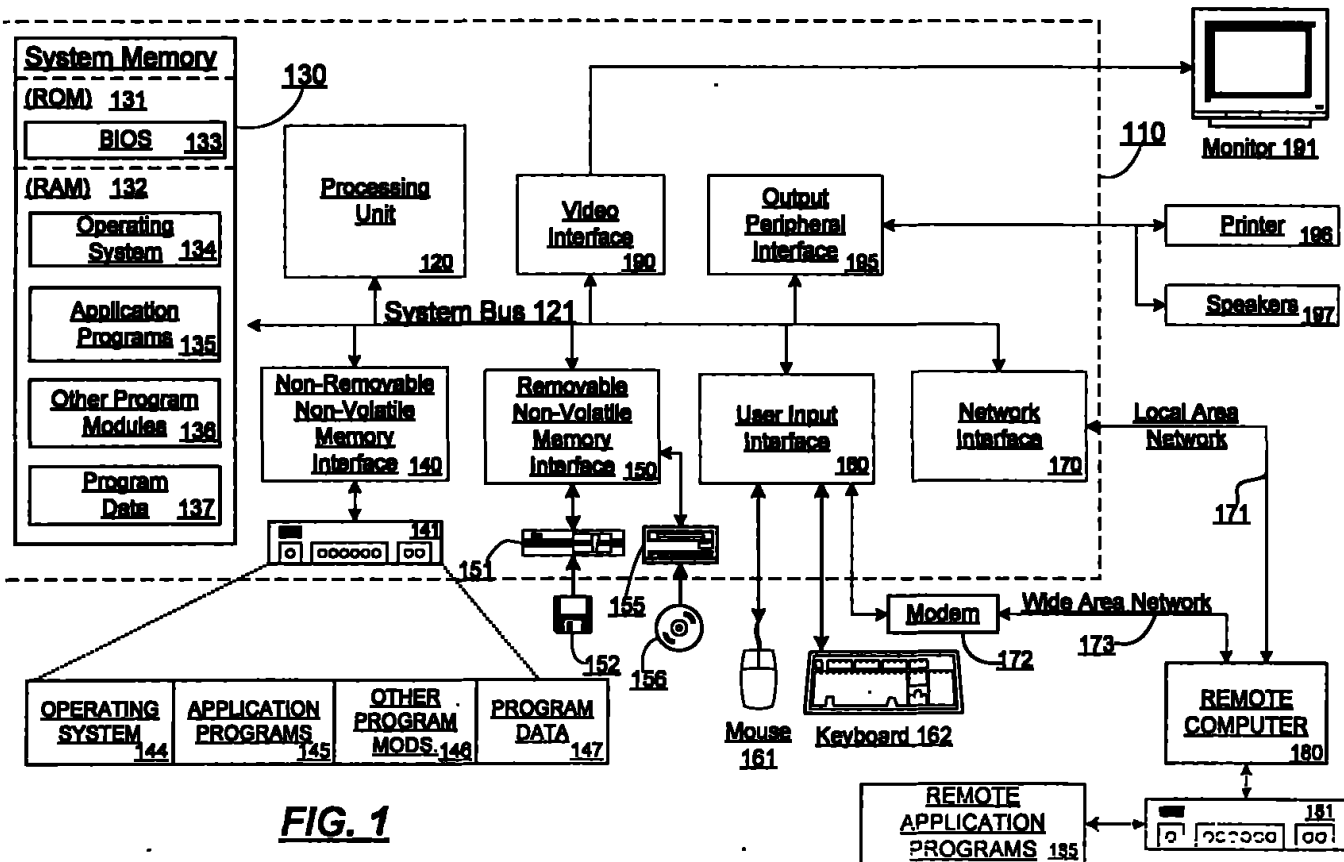
30. The method of claim 23, wherein said machine comprises a virtual machine.

### **ABSTRACT OF THE DISCLOSURE**

A mechanism for protected operating system boot that prevents rogue components from being loaded with the operating system, and thus prevents divulgence of the system key under inappropriate circumstances. After a portion of the machine startup procedure has occurred, the operating system loader is run, the loader is validated, and a correct machine state is either verified to exist and/or created. Once the loader has been verified to be a legitimate loader, and the machine state under which it is running is verified to be correct, the loader's future behavior is known to protect against the loading of rogue components that could cause divulgence of the system key. With the loader's behavior being known to be safe for the system key, the validator may unseal the system key and provides it to the loader.

70

Computing Environment 100



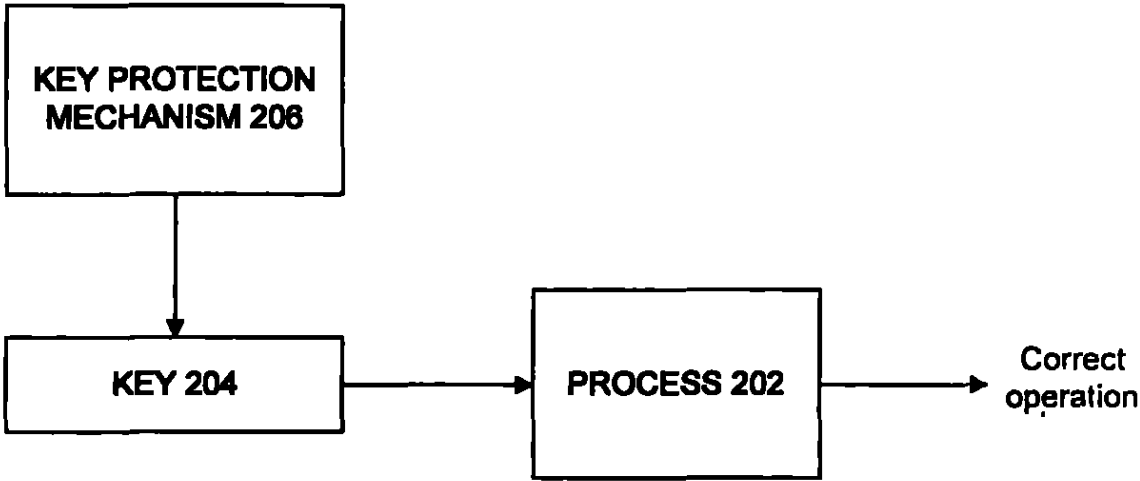


**FIG. 2**

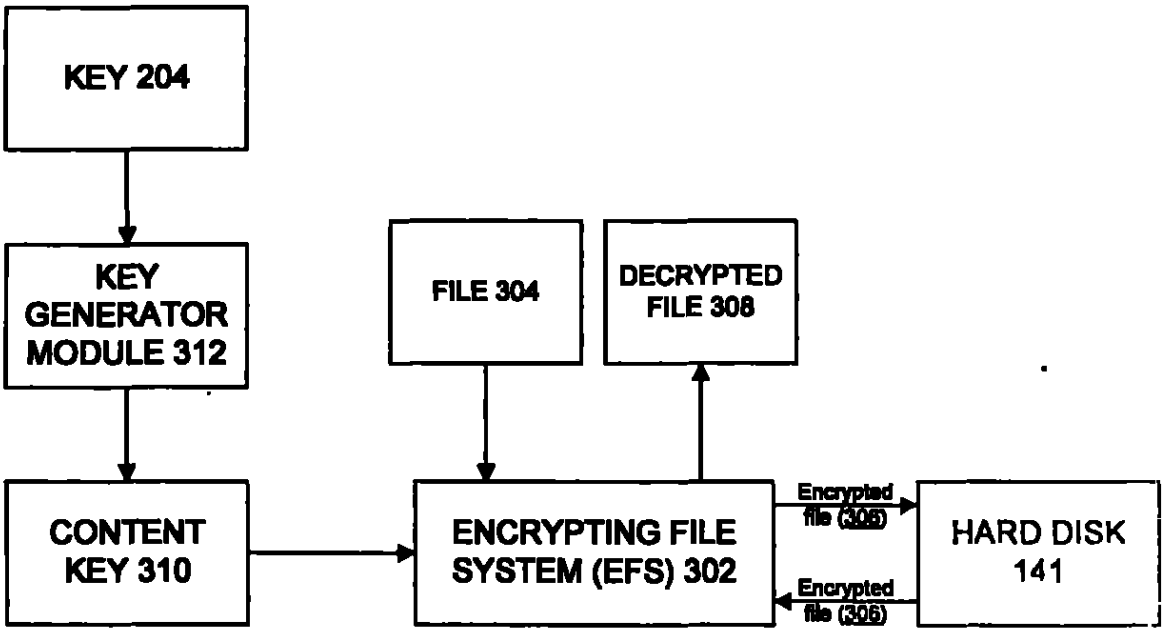


**FIG. 3**

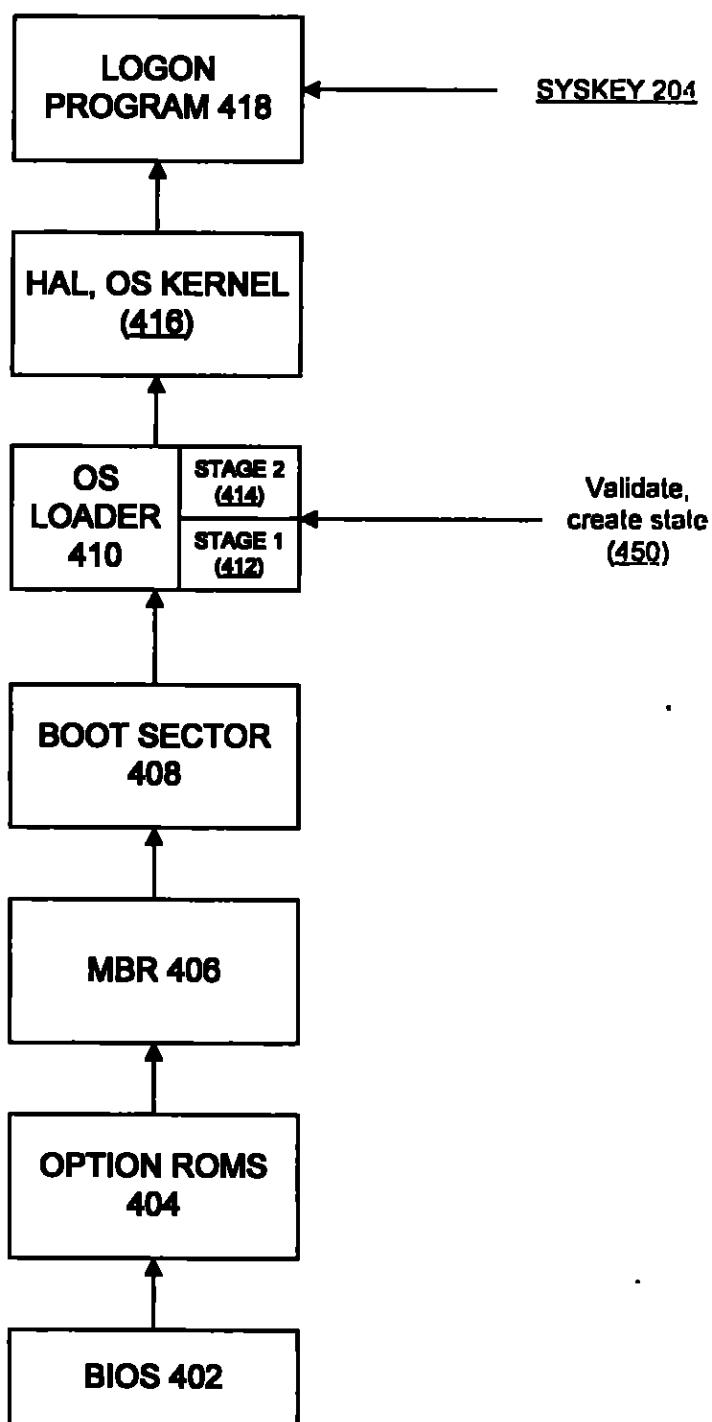


**FIG. 4**

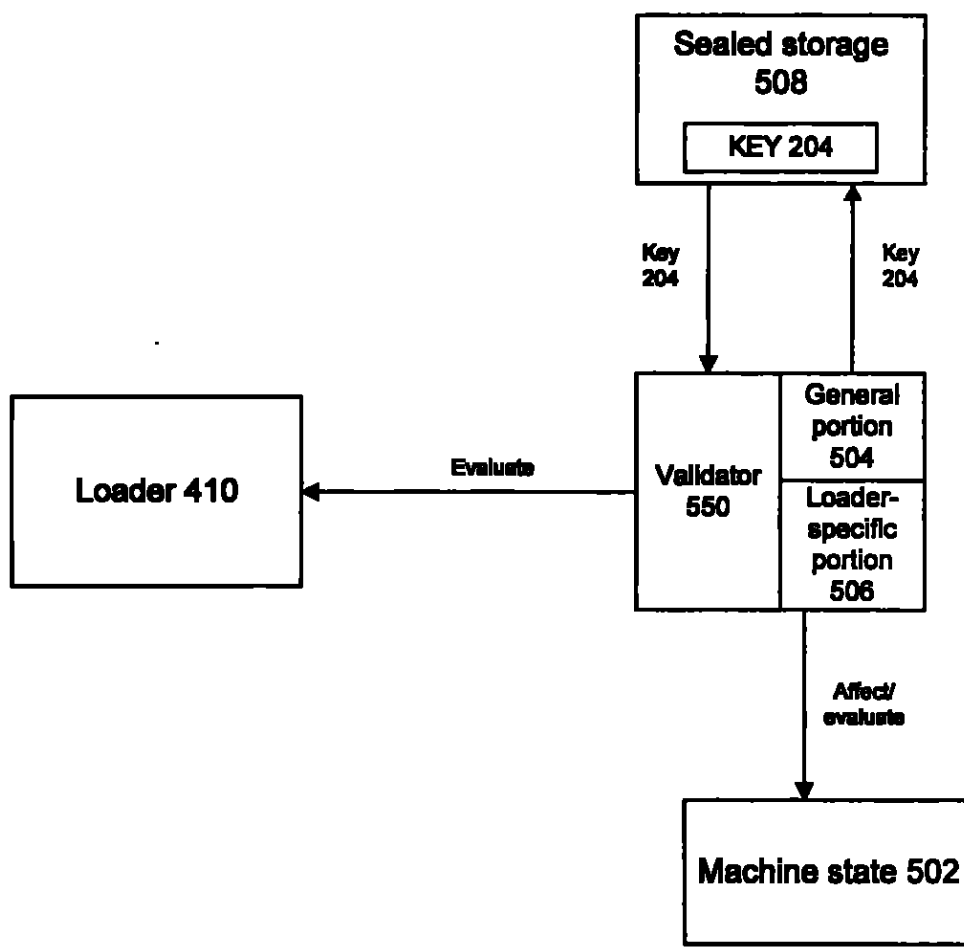
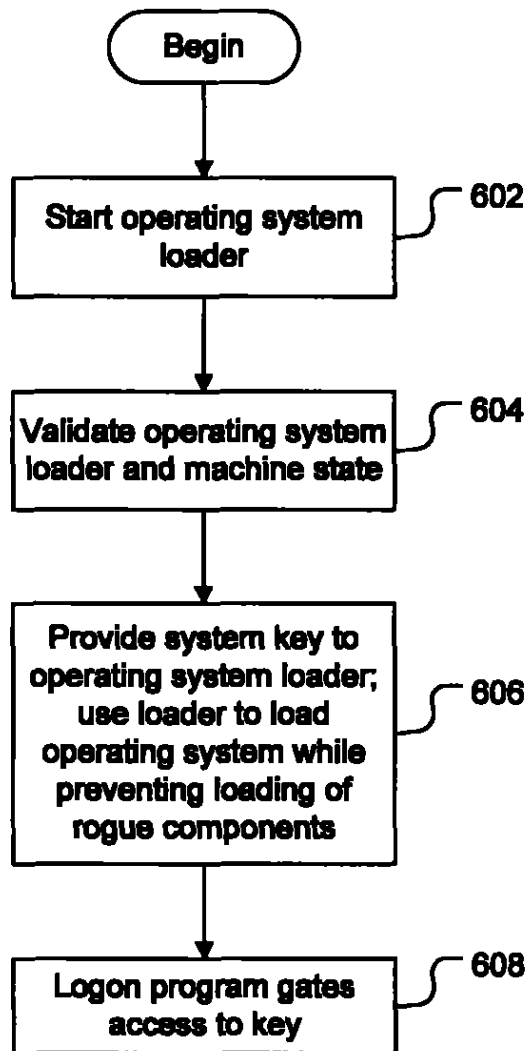


**FIG. 5**

**FIG. 6**

### **ABSTRACT OF THE DISCLOSURE**

A mechanism for protected operating system boot that prevents rogue components from being loaded with the operating system, and thus prevents divulgence of the system key under inappropriate circumstances. After a portion of the machine startup procedure has occurred, the operating system loader is run, the loader is validated, and a correct machine state is either verified to exist and/or created. Once the loader has been verified to be a legitimate loader, and the machine state under which it is running is verified to be correct, the loader's future behavior is known to protect against the loading of rogue components that could cause divulgence of the system key. With the loader's behavior being known to be safe for the system key, the validator may unseal the system key and provides it to the loader.

PA 1295421

# THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

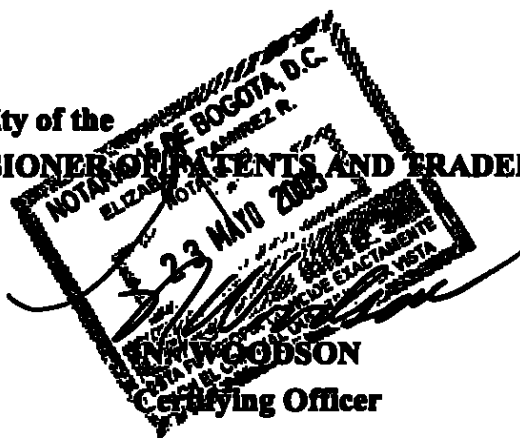
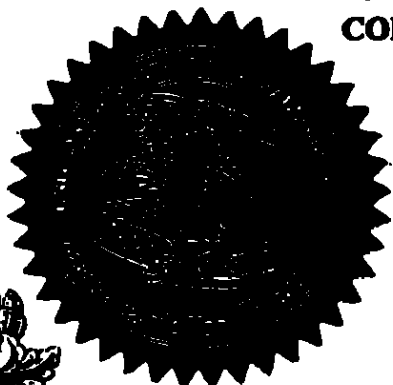
March 16, 2005

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: 10/882,134

FILING DATE: June 30, 2004

By Authority of the  
COMMISSIONER OF PATENTS AND TRADEMARKS



20427 U.S. PTO  
063004

DOCKET NO.: MSFT-3013/305832.01

PATENT

22581 U.S. PTO  
10/882134

063004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:

Bryan Mark Willman, Paul England,

Kenneth D. Ray, Jamie Hunter, Lonny

McMichael, Derek LaSalle, Pierre

Jacomot, Mark Paley, Thekkthalackal

Varugis Kurien, David B. Cross

For: SYSTEM AND METHOD FOR PROTECTED OPERATING SYSTEM BOOT  
USING STATE VALIDATION

EXPRESS MAIL LABEL NO: EL 998514625 US  
DATE OF DEPOSIT: June 30, 2004

EL 998514625US

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

PATENT APPLICATION TRANSMITTAL LETTER

Transmitted herewith for filing, please find:

☒ A Utility Application under 37 CFR § 1.53(b).

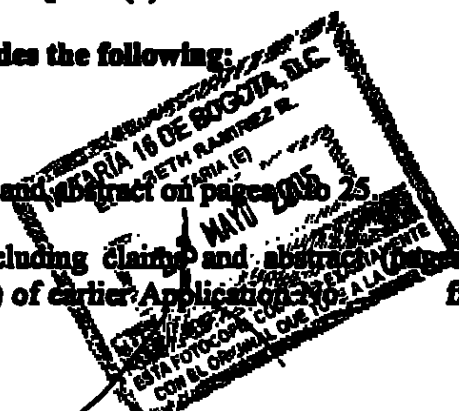
The above-identified application includes the following:

☐ A Request for Nonpublication

☒ A Specification including claims and abstract on pages 1-25

☐ A copy of the specification including claims and abstract, and drawings and appendixes (if any) of earlier Application No. 10/100,000 filed \_\_\_\_\_, to which

Page 1 of 5



DOCKET NO.: MSFT-3013/305832.01

PATENT

no new matter has been added. Such earlier application is hereby incorporated into the present application by reference.

- ☐ A New Unexecuted Declaration or Oath and Power of Attorney
- ☒ A New Executed Declaration or Oath and Power of Attorney for Paul England, Kenneth D. Ray, Jamie Hunter, Lonny Dean McMichael, Mark Eliot Paley, Thekkthalackal Varugis Kurien, David B. Cross (unexecuted as to 3 other inventors).
  - ☐ A copy of the executed oath or declaration filed in the earlier application identified above is attached.
  - ☐ An Associate Power of Attorney executed by \_\_\_\_\_ who is listed in the copy of the executed Declaration and Power of Attorney filed herewith.
  - ☐ A Copy of the Power of Attorney with Revocation filed in prior Application No. \_\_\_\_\_
  - ☐ Signed Statement deleting inventor(s) named in the prior application.
- ☐ This application claims foreign priority under 35 U.S.C. § 119 of Application No. \_\_\_\_\_ filed in \_\_\_\_\_ (Country).
- ☐ A Certified Copy of each of the above applications for which priority is claimed:
  - ☐ is enclosed.
  - ☐ has been filed in prior Application No. \_\_\_\_\_ filed \_\_\_\_\_.
- ☒ 15 Sheets of Formal Drawings and/or Photographs.
  - ☒ Figure 4 should be published.
  - ☐ Petition to Accept Color Photographs is enclosed.
- ☐ This continuation/divisional application is assigned of record to \_\_\_\_\_ as evidenced by the assignment recorded on \_\_\_\_\_ at Reel No. \_\_\_\_\_ and Page No. \_\_\_\_\_
- ☐ A newly Executed Assignment transferring \_\_\_\_\_
  - ☐ A Recordation Form Cover Sheet.
  - ☐ Recordation Fee - \$40.00
- ☐ A Preliminary Amendment is enclosed.
- ☐ A Sequence Listing consisting of pages 1-\_\_\_\_\_

DOCKET NO.: MSFT-3013/305832.01

PATENT

- ☐ Diskette containing Sequence Listing is enclosed.
- ☐ The computer readable form in the above-identified application is identical with that filed in Application Number , filed . In accordance with 37 CFR § 1.821(e), please use the sequence listing filed on and received by the PTO on as the computer readable form filed in that application as the computer readable form for the instant application. It is understood that the Patent and Trademark Office will make the necessary change in application number and filing date for the computer readable form that will be used for the instant application.
- ☐ A paper copy of the Sequence Listing:
  - ☐ was filed in the above-identified parent application on and received by the PTO on
  - ☐ is included herewith in a separately filed preliminary amendment for incorporation into the specification.
- ☐ A Statement to Support Submission of Sequence Information is enclosed.
- ☒ Information Disclosure Statement including:
  - ☒ Form 1449.
  - ☒ Copies of References 1-6 listed on the attached Form PTO-1449.
- ☐ A copy of Petition for Extension of Time as filed in the prior case for copendency.
- ☒ Appended Material as follows: Application Data Sheet.
- ☒ Return Receipt Postcard (should be specifically itemized).



DOCKET NO.: MSFT-3013/305832.01

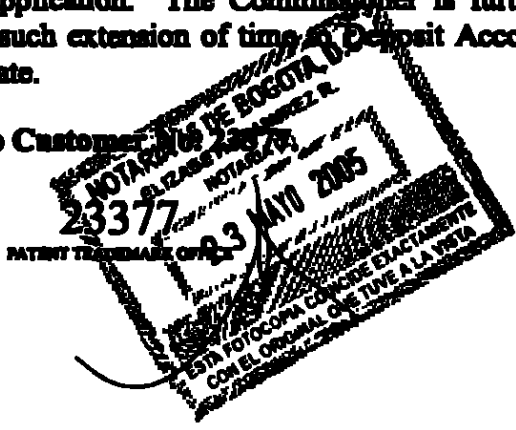
PATENT

FEE CALCULATION:

- ☐ Applicant(s) by its/their undersigned attorney, claims small entity status under 37 CFR § 1.27 as (CHOOSE ONE: an Independent Inventor, a Small Business Concern, or a Nonprofit Organization).

|                                                                          |           |           | SMALL ENTITY |     | NOT SMALL ENTITY |          |
|--------------------------------------------------------------------------|-----------|-----------|--------------|-----|------------------|----------|
|                                                                          |           |           | RATE         | FEE | RATE             | FEE      |
| DESIGN APPLICATION BASE FEE                                              |           |           | \$170.00     |     | \$340.00         |          |
| UTILITY APPLICATION BASE FEE                                             |           |           | \$385.00     |     | \$770.00         | \$770.00 |
| UTILITY APPLICATION; ALL CLAIMS CALCULATED AFTER ENTRY OF ALL AMENDMENTS |           |           |              |     |                  |          |
|                                                                          | No. Filed | No. Extra |              |     |                  |          |
| TOTAL NO. OF CLAIMS                                                      | 30-20 =   | 10        | \$9 each     |     | \$18 each        | \$180.00 |
| NO. OF INDEPENDENT CLAIMS                                                | 3-3 =     | 0         | \$43 each    |     | \$86 each        |          |
| MULTIPLE DEPENDENT CLAIM FEE                                             |           |           | \$145        |     | \$290            |          |
| ADDITIONAL FILING FEE                                                    |           |           |              |     |                  |          |
| TOTAL FILING FEE DUE                                                     |           |           |              |     |                  | \$950.00 |

- ☒ A Check in the amount of \$950.00 is attached. Please charge any deficiency or credit any overpayment to Deposit Account No. 23-3050.
- ☐ Please charge Deposit Account No. 23-3050 in the amount of \$       .00. This sheet is attached in duplicate.
- ☒ The Commissioner is hereby requested to grant an extension of time for the appropriate length of time, should one be necessary, in connection with this filing or any future filing submitted to the U.S. Patent and Trademark Office in the above-identified application during the pendency of this application. The Commissioner is further authorized to charge any fees related to any such extension of time to Deposit Account No. 23-3050. This sheet is provided in duplicate.
- ☒ Address all correspondence to Customer:



**DOCKET NO.: MSFT-3013/305832.01**

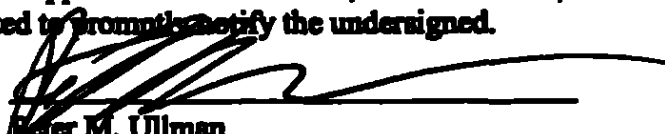
**PATENT**

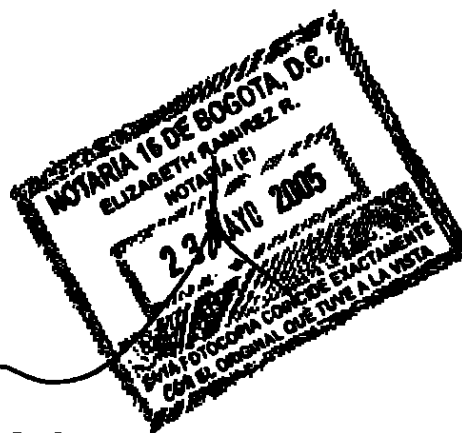
☒ **Direct all telephone calls to:**

**Peter M. Ullman  
WOODCOCK WASHBURN LLP  
One Liberty Place - 46th Floor  
Philadelphia PA 19103  
Telephone No.: (215) 568-3100  
Facsimile No.: (215) 568-3439**

**SHOULD ANY DEFICIENCIES APPEAR** with respect to this application, including deficiencies in payment of fees, missing parts of the application or otherwise, the United States Patent and Trademark Office is respectfully requested to promptly notify the undersigned.

**Date: June 30, 2004**

  
**Peter M. Ullman  
Registration No. 43,963**



**Page 5 of 5**

## FIELD OF THE INVENTION

## BACKGROUND OF THE INVENTION

- 1 -

MSFT-3013/305832.01

not understood. Thus, one aspect of providing security for a computing environment includes verifying that a known program is being used, and that it is proceeding from a known good state.

[0003] One area where predictability of behavior is particularly important is in the loading of an operating system and its components. Although the operating system itself may be designed to provide some level of trustworthiness as to its behavior, the time before such an operating system has been loaded is a time when the system is particularly vulnerable to attack, since the infrastructure that protects the operating system from attacks may not have been established yet (or may be in the process of being established). Thus, ensuring that the operating system loads in a predictable way is important for protecting the operating system from certain classes of attacks.

[0004] One type of security breach that can flow from non-secure loading of an operating system relates to the protection of the key (or keys) that enable certain restricted functionality. By way of example but not limitation, the MICROSOFT WINDOWS operating systems employ a system key, or "SYSKEY," that is used to protect various processes by making the correct performance of those processes dependent on the availability of SYSKEY. For example, the key needed to decrypt private information that is stored by the operating system in encrypted form may be derivable from the SYSKEY.

[0005] Conventionally, the keys needed to perform restricted operations are protected by the logon procedure. Typically, the user must correctly authenticate himself (e.g., by providing correct logon credentials, such as a username/password combination) prior to commencing use of the system. Use of the keys is enabled only if the user correctly authenticates, and the system will only permit the user a limited number of tries (e.g., three) before concluding that the user has failed to logon properly. (This type of limit on the number of attempts to logon prevents unauthorized users from enabling use of protected functionality by using a brute force attack to guess the password in the case of, say, a stolen laptop computer.) However, using the logon procedure to protect access to keys assumes that the operating system loader correctly loaded the operating system with the correct logon program, and that the use of the keys has not been otherwise enabled by rogue code that may be running. If a rogue loader was used instead, and the rogue loader causes a rogue logon program to be loaded with the operating system, then the use of keys might be enabled, or the keys might even be divulged, without the correct credentials having been entered. Since the loading of the operating system provides an opportunity for a security breach, protection

MSFT-3013/305832.01

of the keys in such a situation requires that the loading of the operating system take place under circumstances where it can be verified to take place correctly.

[0006] One problem that occurs with verifying the security of an operating system load process is that legitimate operating system loads can involve many different programs (e.g., there are numerous different "option ROMs," which are pre-OS programs that run during a system boot procedure), and there are numerous different procedures that can be performed as part of an operating system load. Thus, there are a nearly uncountable number of different legitimate machine states during a load, and identifying all such states and verifying that the machine is in a known good state may prove to be an infeasible task. However, not all parts of the load procedure have security implications. It may be more efficient to let the load proceed without any attempt to evaluate its security, but then set the environment to a known good state before starting any procedure that could affect a security-related function, such as the distribution of keys. More generally, an arbitrary system can be allowed to run for some time without any type of security evaluation, as long as the system can be set to a known good state before allowing any actions that have security implications to take place.

[0007] In view of the foregoing, there is a need for a mechanism that overcomes the drawbacks of the prior art.

#### SUMMARY OF THE INVENTION

[0008] The present invention provides for the loading of an operating system under circumstances where the load can be verified to be performed correctly. When a machine is started, the very early startup procedure (e.g., BIOS, option ROMs, master boot record, boot sector, etc.) are performed. After these early procedures are performed, the operating system loader is started, and may perform various preliminary tasks. After the operating system loader is started and has performed such preliminary tasks, a validation of the operating system loader is performed.

[0009] The validation comprises performing validity tests on the loader itself, or a portion of the loader (e.g., checksums, or other tests that are aimed at evaluating the identity and correctness of the loader), as well as evaluating the current machine state (or forcing the machine to conform to a known good state). If the loader (or relevant portion) is known to be correct, and the machine state is one in which the loader has previously been verified to behave correctly, then the behavior of the

MSFT-3013/305832.01

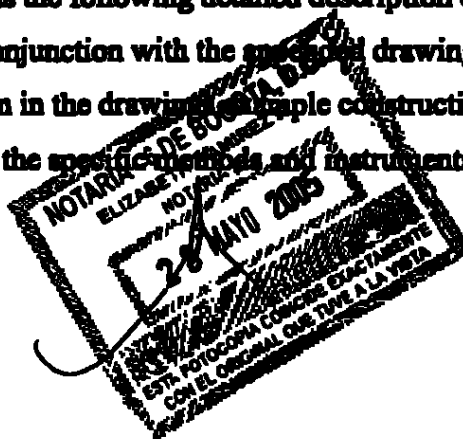
loader can be predicted. Thus, it can be assured that a correct loader operating under a correct machine state will not load a component that would cause data that enables protected functions (e.g., cryptographic keys, such as the system key) to be distributed under inappropriate circumstances.

[0010] Preferably, the validation is performed by a validator that runs in a high-assurance environment. A high assurance environment is one in which it is possible to provide a relatively high degree of assurance that processes performed in it will run in the manner they are expected to. Thus, the assurance that the validator will operate correctly is derived from the fact that the validator is verified by a process that runs in the high-assurance environment, (e.g., by verifying a signature of its binary), and from the underlying trust that processes in the high-assurance environment will be carried out correctly. (Or, at least, there is a level of assurance that the high-assurance environment will not interfere with, or permit interference with, the correct operation of a process that operates within such an environment; one must still have a separate basis of trust that the program that implements a process within the high-assurance environment will correctly perform in the manner that it is expected to perform.) A high-assurance environment may provide sealed storage, which is a storage facility in which data can be sealed to a particular object (e.g., to a particular program), and that the high-assurance environment control in such a way that the sealed data will not be released to any object other than the one to which the data is sealed (as verified by the high-assurance environment). The validator can use this sealed storage to seal a key (e.g., the SYSKEY) to itself, and can refuse to unseal the key except to the correct object, and when the circumstances (e.g., the machine state) are satisfactory under some standard.

[0011] Other features of the invention are described below.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the accompanying drawings. For the purpose of illustrating the invention, there is shown in the drawings a number of constructions of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:



MSFT-3013/305832.01

[0013] FIG. 1 is a block diagram of an example computing environment in which aspects of the invention may be implemented;

[0014] FIG. 2 is a block diagram of a system that employs a process whose correct operation is dependent on a system key;

[0015] FIG. 3 is a block diagram of an encrypted file system that protects encrypted data from unauthorized decryption by making decryption dependent on a system key;

[0016] FIG. 4 is a flow diagram of an example boot process with validation in accordance with aspects of the invention;

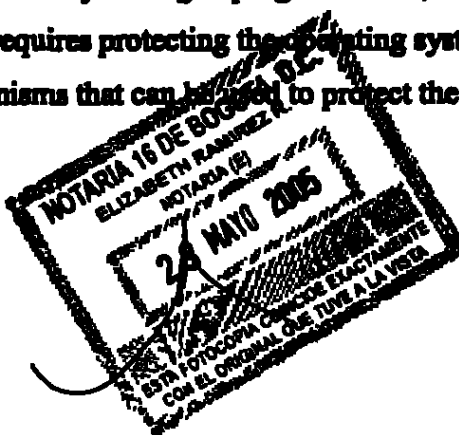
[0017] FIG. 5 is a block diagram of an example validator in accordance with aspects of the invention; and

[0018] FIG. 6 is a flow diagram of an example process of protecting a system key in accordance with aspects of the invention.

## DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

### Overview

[0019] Several processes that may take place under an operating system are dependent on one or more keys for their correct operation. Access to the keys may be controlled by an authentication program, such as a logon program, which refuses to enable use of the key(s) unless the user provides correct credentials, such as a username/password combination. Thus, by the logon program's refusal to enable the use of keys in the absence of correct credentials, several processes (e.g., decryption of encrypted files) can be disrupted (or entirely prevented) for users who do not know the password. While the logon program may be effective at gating access to the keys, an operating system loader could be tricked into loading a different component that would distribute the keys outside of the authentication rules imposed by the logon program. Thus, when keys are distributed in this manner, protecting the keys requires protecting the operating system loading process. The present invention provides mechanisms that can be used to protect the loading process.



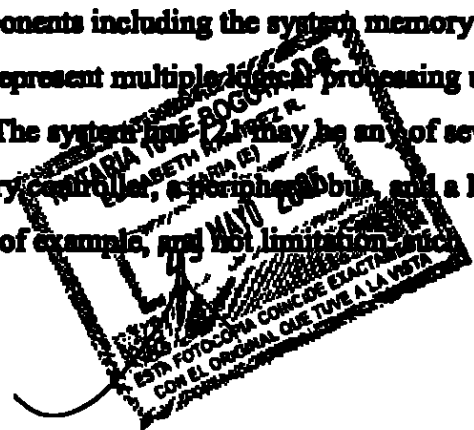
**Exemplary Computing Arrangement**

[0020] FIG. 1 shows an exemplary computing environment in which aspects of the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0021] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, embedded systems, distributed computing environments that include any of the above systems or devices, and the like.

[0022] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules and other data may be located in both local and remote computer storage media including memory storage devices.

[0023] With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The processing unit 120 may represent multiple identical processing units such as those supported on a multi-threaded processor. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation,

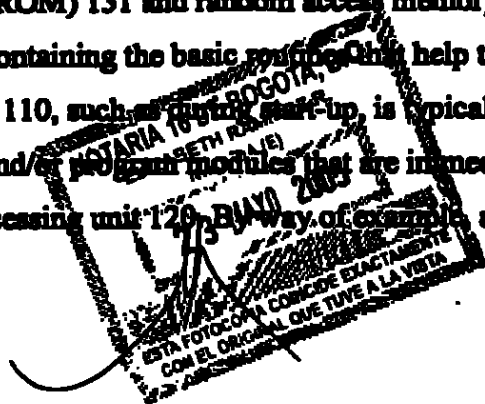


MSFT-3013/305832.01

architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus). The system bus 121 may also be implemented as a point-to-point connection, switching fabric, or the like, among the communicating devices.

[0024] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0025] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and

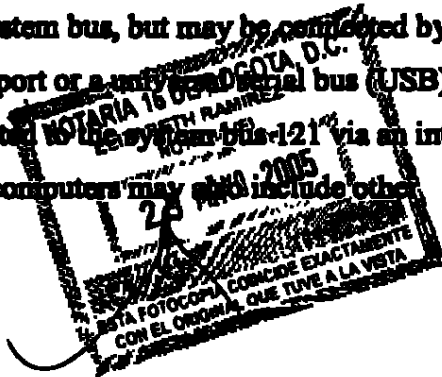


MSFT-3013/305832.01

not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0026] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0027] The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other



MSFT-3013/305832.01

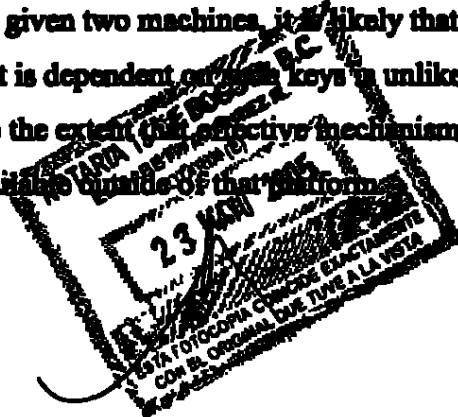
peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0028] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0029] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

#### Action protected by key

[0030] A computing environment may employ a key on which certain processes that take place within the environment depend for their correct operation. The system key, or "SYSKEY" used by the MICROSOFT WINDOWS operating systems is one example of such a key, but is not a limiting example. In a preferred embodiment, a key on which a process depends is a unique, cryptographically random per-platform key – i.e., given two machines, it is likely that the two machines have different keys. Thus, a process that is dependent on such keys is unlikely to be portable from one platform to another – at least to the extent that effective mechanisms are employed to ensure that a platform's key is unavailable outside of that platform.



MSFT-3013/305832.01

**[0031] FIG. 2 shows an example system in which a process runs that is dependent on a key. Process 202 is dependent on key 204 in order to operate correctly. It should be noted that process 202 is not limited to the traditional notion of a process – i.e., a unit of execution that can be managed by an operating system and assigned an address space – but rather refers more generally to any operation or series of operations that can be performed on a computer. It should be further noted that, while this example shows a process that is dependent on a cryptographic key, the term “process” as used herein is not limited to processes that perform cryptographic operations.**

**[0032] As shown in FIG. 2, if key 204 is available as an input to process 202, then process 202 operates correctly. On the other hand, if key 204 is not available as an input to process 202, then process 202 does not operate correctly. A key protection mechanism 206 controls access to key 204 – i.e., mechanism 206 either provides, or does not provide, process 202 with key 204, depending on whether the relevant security conditions have been met. For example, a user may have to log in and provide a correct password before mechanism 206 will enable use of key 204.**

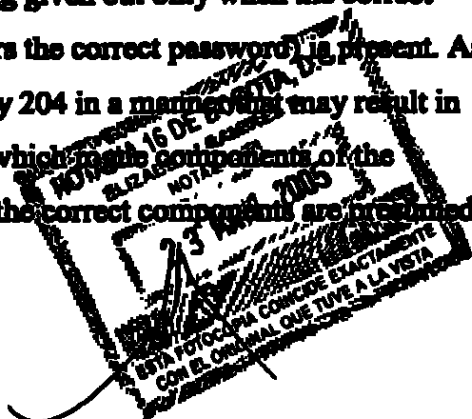
**{0033} It should be noted that the denial of key 204 in order to prevent process 202 from operating correctly is sometimes a desired result. For example, the encryption/decryption of files is one example of a process 202 that is protected by key 204. The correct decryption of a file may be dependent on access to key 204. If the user cannot correctly log in and authenticate himself, then it may be desirable for the decryption of a file not to proceed, since the inability of a user to log in may indicate that the computer is being operated by someone other than its intended user (e.g., in the case of a stolen laptop). Thus, key protection mechanism 206 can make access to key 204 dependent on the relevant security conditions being met, and can use denial of key 204 to disrupt processes that need to be disrupted when those security conditions are not met. Mechanism 206's ability to disrupt processes in this manner is dependent on those processes' requiring key 204 for their correct operation, since it is this dependency that enables the act of denying key 204 to disrupt the process.**

**[0034]** FIG. 3 shows a specific (but non-limiting) example of a process that is dependent on key 204. In the example of FIG. 3, the example process is an Encrypting File System (EFS) 302 that stores files in encrypted form, and also decrypts the encrypted files. It should be noted that one purpose of such file encryption is to protect data on a laptop computer from being recovered by a thief in the event that the laptop is stolen. When a file 304 is generated for storage, the file is

provided to EFS 302. EFS 302 then encrypts file 304, and turns file 304 into encrypted file 306, which is stored on hard disk 141. When a request is made to retrieve encrypted file 306, EFS 302 retrieves encrypted file 306 and decrypts it to generate decrypted file 308. (In practice, the substance of decrypted file 308 is the same as that of original file 304, although for clarity FIG. 3 shows these two instances of the file separately: file 304 is the original file, and decrypted file 308 is the same file after it has been encrypted, stored, retrieved, and decrypted by EFS 302.)

[0035] It will be noted that one input to EFS 302 is content key 310. Content key 310 is preferably a symmetric key that functions as an input to a cryptographic method. Content key 310 is used to encrypt file 304 in order to create encrypted file 306, and is also used to decrypt encrypted file 306 in order to create decrypted file 308. It can be appreciated that storing key 310 in some easily retrievable place could quickly render EFS 302's ability to protect the data ineffective: If the content key is easily available on a hard disk, or if it is easily derivable from some known feature of the laptop (e.g., the processor serial number), then it would make no difference whether the files were stored in encrypted form, since a thief could easily find the key and decrypt them. Thus, it is desirable to protect the key by deriving it in some way that can only be done with the cooperation of the true owner. One way to protect the key is to use a key generator module 312, which receives key 204 as input, and derives content key 310 as a function of key 204. Thus, to the extent that key 204 will only be provided under appropriately secure circumstances, it is likewise true that content key 310 will only be derivable under appropriate circumstances. In other words, by making the derivation of content key 310 dependent on the availability of key 204, whatever protection is afforded to key 204 can be extended to content key 310. For example, if providing key 204 requires that the user log in by providing a correct password, then it can be assured that content key 310 will not be available unless the user correctly logs in.

[0036] Thus, protecting key 204 from being given out under the wrong circumstances is important, since other processes may depend on key 204 being given out only when the correct security context (e.g., a legitimate, logged in user, who tenders the correct password) is present. As described below, one way to cause a machine to distribute key 204 in a manner that may result in misuse is to boot the machine in an insecure environment in which the components of the operating system can replace the correct components (where the correct components are presumed



to be able to protect key 204). Thus, mechanisms are described below that ensure that a machine is booted in a known (safe) environment before key 204 can be distributed.

#### Boot Process with State Validation

[0037] FIG. 4 shows a typical sequence of events that are used in a boot procedure.

[0038] Initially, the machine is powered on. A typical machine is configured to begin executing instructions at some fixed address at the time that it is powered on. The instructions typically contained at this address are known as a "BIOS" 402, or "Basic Input/Output System." At the conclusion of BIOS 402's execution, BIOS 402 starts execution of small programs called "option ROMs" 404. Option ROMs are programs that perform very basic early-boot functions, such as setting the hardware password for the machine, or selecting which of several operating systems is to be booted. After option ROMs 404 execute, the machine is instructed to load the Master Boot Record (MBR) 406.

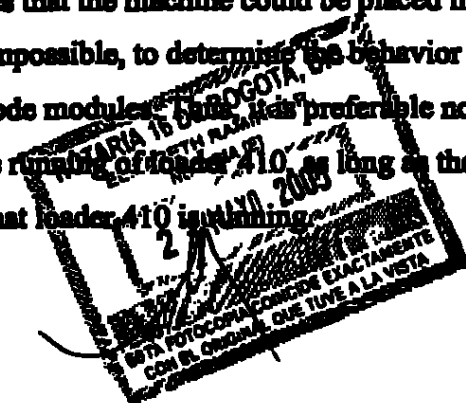
[0039] MBR 406 is an executable program. Typically, MBR 406 resides on the first sector of a computer's hard disk, and begins by looking up which partition to use for further booting in the partition table. (For example, a disk may be partitioned for use with different operating systems, and each operating system may require a different boot procedure.) After the correct partition has been looked up, MBR 406 transfers control to the boot sector 408 associated with that partition. Boot sector then commences the process of loading the operating system loader 410 that will ultimately load the operating system. It should be noted that MBR 406 is shown in FIG. 4 merely to show how such a component would fit into an example boot process, and the invention is not limited to boot procedures that use MBR 406.

[0040] During the time that operating system loader 410 is running, the loader and machine state are validated (450). ("Machine," in this context, can refer to either a physical machine or a virtual machine.) The validation is performed by a trusted software component that operates in a high-assurance environment (an embodiment of which is more particularly described below), and therefore there is some degree of assurance/trustworthiness that the validation of loader 410 is being done correctly. In essence, by validating the relevant aspects of the machine state while operating system loader 410 has control of the machine, and by validating loader 410, it is possible to provide some level of assurance that illegitimate or rogue versions of operating system components (e.g., the

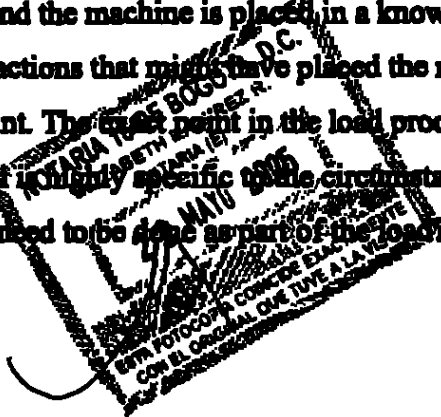
MSFT-3013/305832.01

hardware abstraction layer, the kernel, drivers, etc. (416)) will not be loaded later on, up to the time that logon program 418 is run. Preventing rogue components from being loaded up to the time that logon program 418 is run is important because one of the first things that the operating system will do after it is loaded is run a logon program 418 that gates access to key 204, and if rogue components could be loaded, then these components could cause logon program 418 to misbehave in a manner that could result in distributing key 204 under inappropriate circumstances, thereby compromising the security of all components that depend on the protection of key 204 as described above. Thus, protection of key 204 can be achieved by exerting tight control over the state of the machine from the time that the operating system loader 410 is run until the logon program has completed.

[0041] Validation 450 takes place during the time that operating system loader 410 is running. Validation includes validating the loader and the machine state, and may also include setting the machine state to a known good state. The basic idea behind validating and setting the machine state is to place the machine in a state where, if the loader executes while the machine is in that state, then the loader will not load any rogue components or otherwise behave in ways that would result in security breaches. Validation 450 ensures that the loader code is, in fact, the code that has previously been verified to behave correctly, and also ensures that the machine is in a state under which this known code will behave correctly (either by verifying that the machine already is in that state, or by placing the machine in that state). It will be appreciated that this technique is able to leverage the existing (safe) state of the machine to ensure that the future states of the machine will also be safe, and this leveraging is made possible by the fact that the behavior of the operating system loader 410 is not only known and understood, but also tightly circumscribed. Based on this observation, it will be appreciated that validation 450 should not take place too early (e.g., at the time that option ROMs 404, or MBR 406, is executed), since the wide variety of code from numerous different sources, and the vast breadth of states that the machine could be placed in as a result of executing such code, makes it difficult, if not impossible, to determine the behavior of the machine while executing all of these various different code modules. It is preferable not to be concerned with what states the machine is in prior to the running of loader 410, as long as the machine can be placed in a legitimate state at the time that loader 410 is running.



[0042] In a preferred embodiment, the execution of loader 410 is divided into two stages: stage 1 (412) and stage 2 (414). Preferably, before stage 2 is entered, the code that implements stage 2 is validated (450), and the trusted code that performs validation 450 then jumps into stage 2 at a well-defined entry point. (The "trusted code" that performs validation is the above-mentioned program that runs in a high-assurance environment. It will be understood by those of skill in the art that "trusted" does not imply absolute infallibility, but merely means that there is some basis to assume that the code will do its job correctly. Since the behavior of a program can be affected by the environment in which it runs, executing the trusted code in a high-assurance environment means that the trusted code will operate correctly: (1) the trusted code can be trusted to perform its function correctly in an environment that conforms to some expectations, and (2) the high-assurance environment can be trusted to correctly provide an environment that conforms to such expectations.) Stage 2 may then validate (according to some defined standard) any information that it has received from stage 1. The dividing line between stage 1 and stage 2 reflects the fact that there may be some aspects of a boot procedure that can be performed without any security verification, and without having any consequences for some security-related task (such as the distribution of key 204), as long as the relevant programs and state can be validated at some point before events could be set in motion that could cause such a security-related task not to be performed correctly. Moreover, there may be problems with attempting to validate state too early in the boot process: the actual execution path that a legitimate boot procedure can take is quite variable, so it becomes difficult to define the difference between valid or invalid machine states under such variable circumstances. It may make sense to simply allow the load to proceed through any of these variable states, without trying to determine whether any state that the machine passes through is valid. Stage 1 thus represents the portion of the load that proceeds without any validation. At the end of stage 1, the loader and machine state are validated, and stage 2 begins. The presumption of such a dividing line is that processes can execute without any security restrictions, up to a certain point – at which point all relevant factors about the machine are validated and the machine is placed in a known good state, which essentially nullifies the effect of any prior actions that might have placed the machine in a state that is unacceptable from a security standpoint. The first point in the load process where the dividing line between stage 1 and stage 2 is drawn is highly specific to the circumstances (e.g., What does the loader code look like? What tasks need to be done as part of the load?), and



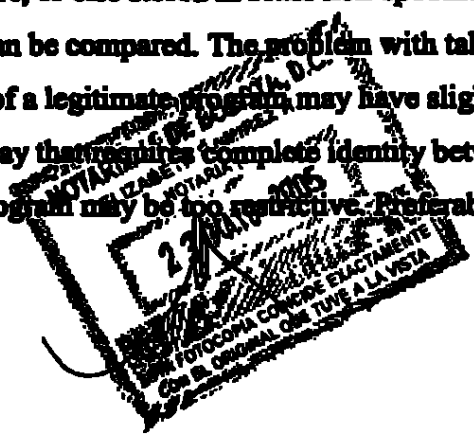
MSFT-3013/305832.01

represents a sort of tradeoff: On the one hand, the dividing line should be late enough that, for future actions, the amount of variation in legitimate load behavior is small enough that legitimate behavior can feasibly be distinguished from illegitimate behavior. (As noted above, in the early load procedure, the large number of option ROMs and other variables make the number of possible execution paths so great that it is difficult to distinguish legitimate behaviors from illegitimate behaviors.) On the other hand, the dividing line should be early enough that it precedes any events (e.g., the loading of the logon program) that could have an affect on security, such as causing keys to be distributed inappropriately. In general, the dividing line between stage 1 and stage 2 allows one to run an arbitrary system in a "raw" (or "open," or "unvalidated") state for some time, and then start whatever components are necessary to verify behavior (e.g., a high-assurance environment), and then use such components to validate the current state of the machine (or force the machine into a known good state), at which point the process of doing something that can later affect security (e.g., the distribution of keys) can be allowed to proceed.

Validation 450

[0043] Validation 450 is, in essence, the act of verifying that the operating system loader 410 (or, more specifically, in a preferred embodiment, verifying that stage 2 of the loader) is a known trusted program, and ensuring that the machine on which it is running is in an known good state. Thus, validation has two parts: (1) examining the loader program (or stage 2 of the loader program) to ensure that it is the known, trusted program that it is believed to be, and (2) changing the relevant machine state to a known good state under which the loader is known to behave correctly. The premise of performing (1) and (2) is that a known program, operating on a known machine state, will behave in a known way.

[0044] The first part of the validation – i.e., examining the loader – can be performed in a variety of ways. In the simplest example, there may be a hash of the known program (which may be cryptographically signed to create a digital signature, or else stored in some non-spoofable way), against which the actual program that is running can be compared. The problem with taking a hash over the entire program is that different instances of a legitimate program may have slightly different images, so a hash that is computed in a way that requires complete identity between the known program and the running instance of the program may be too restrictive. Preferably,

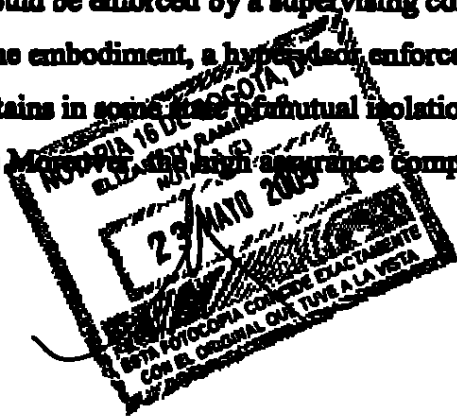


MSFT-3013/305832.01

validation is performed in a way that ensures that the program is what it is supposed to be, but that is not overly restrictive – e.g., the validation process may compare hashes of program components that are known to be fixed, and may perform other tests on program components that may change over time. The component that performs the validation should be tailored to the specifics of the program that is being validated; the invention is not limited to any particular technique for validating a program.

[0045] The second part of the validation – i.e., verifying/setting the machine state – is preferably performed by setting all of the relevant “sources” of state to known values. Typically, the relevant state that can affect an operating program comes from three sources: the CPU, the chipset, and the memory. Thus, these items can be put into a known good state at the time of validation – e.g., the CPU can be put into a known state – e.g., ring 0, with the program counter pointed at a known location, all data registers set to zero, and all memory other than the memory in which the loader program is stored being set to zero. If the loader has been verified to behave correctly when it runs with the machine in this state, then the combination of examining the loader and placing the machine in the known good state should ensure correct behavior up to the time that logon program 418 is run. As discussed above, one of the benefits (although not the only benefit) of correct behavior up to the running of logon program 418 is that rogue components will not be loaded that would cause key 204 to be distributed under inappropriate circumstances.

[0046] It should be noted that, in order for the validation process to protect the load of the operating system from rogue components, there should be sufficient assurance that the validation process itself is being performed correctly. Correctness of the validation process can be ensured by embodying the validator as a trusted agent that runs on or in a high-assurance computing environment. For example, there may be a small, operating system that performs a limited set of functions, but which provides a high degree of assurance that such an operating system will perform according to its specifications. Such an operating system could be run together with other operating systems on a single machine, and the isolation of such a high assurance operating system from the other (less secure) environments on the system could be enforced by a supervising component, such as a hypervisor or virtual machine monitor. (In one embodiment, a hypervisor enforces partitions, which are environments that the hypervisor maintains in some state of mutual isolation from each other, and in which an operating system can run.) Moreover, the high assurance component may



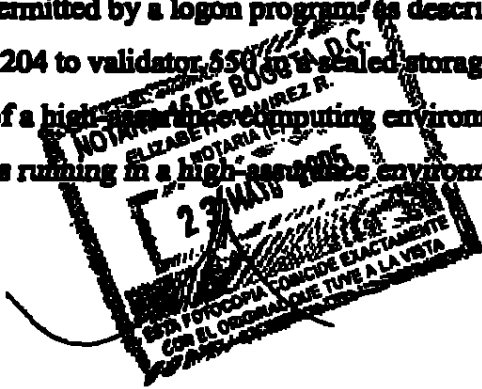
MSFT-3013/305832.01

have exclusive access to a root of trust, such as a hardware module that applies, and strongly protects, per-platform cryptographic keys. The validator could be a program (or "agent") that runs in such a high assurance environment, which would provide assurance that -- to the extent that the high-assurance environment itself can be trusted to resist attacks-- the validator would not be subject to attacks from outside the high-assurance environment that would cause it to behave incorrectly. Additionally, as noted above the high-assurance environment could provide sealed storage (i.e., the ability to store some datum and to release that datum only to a particular object to which the datum has been sealed), and the validator could use this sealed storage to store any keys that are to be distributed. It should be noted that the validator could be an integral part of the high-assurance environment, but could also be a separate component that runs in the high-assurance environment.

[0047] FIG. 5 shows an example of a validator 550 that performs validation in accordance with the procedure described above. Validator 550 has the ability to evaluate loader 410 for correctness and/or compliance with some set of known standards. Additionally, validator 550 has the ability to evaluate and/or affect machine state 502. Using the combination of these abilities, validator 550 can ensure that loader 410 will behave correctly, by ensuring that loader 410 is the loader that is expected to be running, and by ensuring that the machine on which it is running is in a state in which loader 410 is known to behave correctly.

[0048] Structurally, validator 550 may comprise two components: a general portion 504, and a loader-specific portion 506. "General portion" 504 comprises code that is common to a wide variety of validators (or all validators). Loader-specific portion 506 is code that specifically relates to the validation of a particular loader 410 -- i.e., the code that understands what a correct instance of loader 410 (or stage 2 of loader 410) would look like, and that performs tests to ensure that loader 410 conforms to this understanding. Thus, loader-specific portion 506 can be combined with general portion 504 to form a complete validator.

[0049] Validator 550 is the component that ultimately decides whether key 204 will be provided to the loader (which will then later provide it to the operating system, for use by the operating system in accordance with the manner permitted by a logon program, as described above). Validator 550 may protect key 204 by sealing key 204 to validator 550 in a sealed storage facility 508. Sealed storage facility 508 may be a feature of a high-assurance computing environment. Sealed storage facility 508 may permit components running in a high-assurance environment to seal



MSFT-3013/305832.01

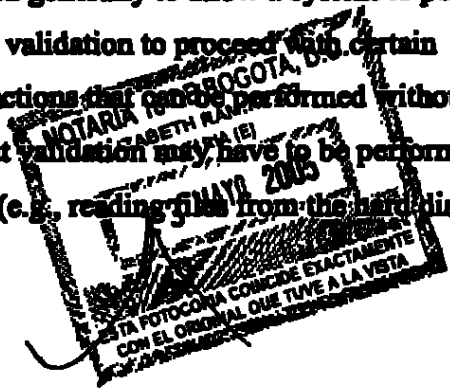
arbitrary data to themselves, such that no other component can retrieve the data. For example, the loader-specific portion 506 of validator 550 may seal key 204 to itself. Sealed storage facility 508 will prevent any component other than loader-specific portion 506 from unsealing the key 204. Thus, key 204 is protected because it can only be obtained from the loader-specific portion 506, and loader-specific portion 506 will only distribute key 204 to a loader that loader-specific portion 506 believes will not cause key 204 to be distributed inappropriately. In another embodiment, the key is sealed based on a combination of components that includes the validator and stage 2 of loader 410. In such an embodiment, the loader-specific portion of the validator 506 can be eliminated, because the sealed storage mechanism itself essentially ensures the presence of a correct stage 2 loader before the key can be unsealed, since the binary for a known correct stage 2 loader is part of the metric that goes into the seal.

Example Process of Protecting SYSKEY

[0050] FIG. 6 shows an example process of protecting a key in accordance with the aspects of the invention described above. At some point after a machine is started, an operating system loader is executed (602). During the execution of the operating system loader, the loader and machine state are validated (604), thereby ensuring that the loader will behave in a predictable manner for the reasons described above. After the loader and machine state have been validated, the loader is used to load the operating system while preventing rogue components from being loaded (606). (As discussed above, validation of the loader and machine state means that the future behavior of the loader is known not to result in the loading of rogue components.) After the operating system is loaded, the logon program is run, which gates access to the key.

Other Example Embodiments

[0051] It should be noted that the mechanisms of the present invention can be used not merely to perform an operating system load, but can be used generally to allow a system to perform some function in a raw (unvalidated) state, while requiring validation to proceed with certain functions. For example, a computer may perform some functions that can be performed without any sort of validation (e.g., the computer can act as a radio), but validation may have to be performed before the computer can perform more sensitive functions (e.g., reading files from the hard disk). In



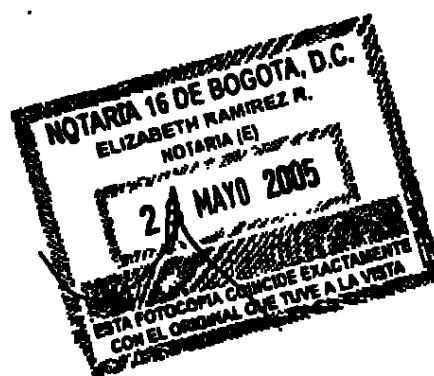
general, a machine can be viewed as both an appliance and a general PC, and that the appliance part doesn't require validation/logon, but the general PC part does.

[0052] Moreover, the validator is not limited to either releasing/not releasing a particular set of keys to the logon program, but could more generally be configured to release specific sets of keys to particular software stacks – e.g., there is one software stack that can get a first key, but validator might release a second key only to a “more heavily validated” software stack.

[0053] Additionally, keys do not necessarily need to be released through a logon procedure, but rather can be provided through arbitrary validation procedures. For example, a DVD player binary could get, say, the DVD function keys after the validator has determined that the player is the correct binary, but without requiring a logon.

[0054] Additionally, some programs could work in some form without validation, but then require a type of validation before being permitted to expose certain functionality. For example, telephony application could start with no logon required, but may require a logon before the keys can be distributed that would permit cryptographic functions of the application to work.

[0055] It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the invention has been described with reference to various embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitations. Further, although the invention has been described herein with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed herein; rather, the invention extends to all functionally equivalent structures, methods and uses, such as are within the scope of the appended claims. Those skilled in the art, having the benefit of the teachings of this specification, may effect numerous modifications thereto and changes may be made without departing from the scope and spirit of the invention in its aspects.



101  
MSFT-3013/305832.01

**What is Claimed:**

1. A computer-readable medium encoded with computer executable instructions to perform a method comprising:

starting an operating system loader;  
validating the identity or correctness of said loader;  
ensuring that a machine on which said operating system loader runs is in a known state; and

if the identity or correctness of said loader validates, and if said machine on which said operating system runs is in a known state, then:

providing a key to said loader; and  
allowing said loader to load an operating system.

2. The computer-readable medium of claim 1, wherein the method further comprises:  
running a logon program that gates access to said key.

3. The computer-readable medium of claim 2, wherein said logon program ensures that a user provides credentials as a condition to said logon program granting access to said key.

4. The computer-readable medium of claim 1, wherein said key is usable to decrypt an encrypted operating system partition.

5. The computer-readable medium of claim 1, wherein said act of validating the identity or correctness of said loader, and said act of ensuring that a machine is in a known state, is performed after all of the following have executed:

a basic input output system;  
an option ROM;  
a master boot record; and  
a boot sector.

- 20 -



MSFT-3013/305832.01

6. The computer-readable medium of claim 5, wherein said act of validating the identity or correctness of said loader, and said act of ensuring that a machine is in a known state, are performed after a portion of said operating system loader has executed.

7. The computer-readable medium of claim 6, wherein said act of validating is performed after all or substantially all of an operating system has started, or after some number of partitions have started, each of said partitions comprising an environment from which some degree of isolation from other partitions is maintained by a hypervisor.

8. The computer-readable medium of claim 1, wherein said key is sealed to a validator that performs said validating act and said ensuring act, and wherein the method further comprises:  
said validator unsealing said key.

9. The computer-readable medium of claim 8, wherein said key is sealed to at least said validator, and wherein said validator validates at least a portion of said loader.

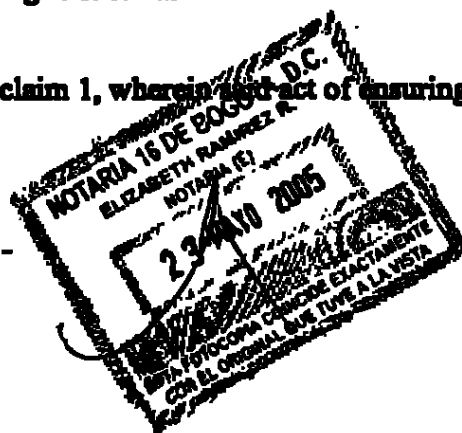
10. The computer-readable medium of claim 8, wherein said key is sealed to said validator and to at least a portion of said loader.

11. The computer-readable medium of claim 1, wherein said machine comprises a physical machine.

12. The computer-readable medium of claim 1, wherein said machine comprises a virtual machine.

13. The computer-readable medium of claim 12, wherein said key is revealed only if an architecture of said virtual machine has not changed or is valid.

14. The computer-readable medium of claim 1, wherein said act of ensuring that a machine is in a known state comprises:



MSFT-3013/305832.01

evaluating a current state of said machine and comparing said current state with said known state.

15. The computer-readable medium of claim 1, wherein said act of ensuring that a machine is in a known state comprises:

setting a current state of said machine to be consistent with said known state.

16. The computer-readable medium of claim 1, wherein said loader is an entire operating system, or an entire instance of an operating system.

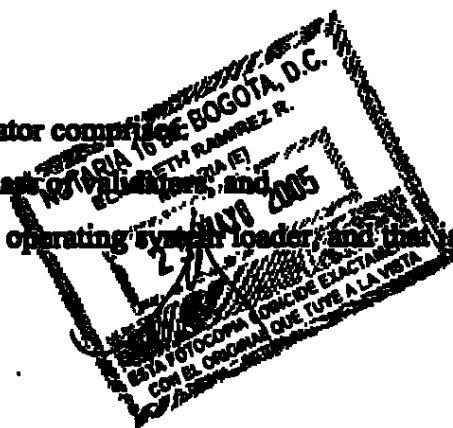
17. A system for performing a boot of an operating system under circumstances that provide assurances as to the reliability of the boot, the system comprising:

a validator that evaluates the correctness or identity of an operating system loader that will load the operating system, and that further evaluates a state of a machine on which said operating system loader will operate, which either allows or disallows the operating system loader to proceed with loading the operating system depending on whether or not the correctness or identity of the operating system loader is verified, and which places said machine in a known state prior to allowing said operating system loader to proceed.

18. The system of claim 17, wherein a key is sealed to said validator, and wherein said validator unseals said key and provides said key to said operating system loader if said operating system loader is allowed to proceed.

19. The system of claim 17, wherein said validator evaluates the correctness or identity of the operating system loader, and evaluates the state of the machine, after the operating system loader has started operation, but before the operating system loader has initialized a kernel or a device driver.

20. The system of claim 17, wherein said validator comprises  
a general portion that is common to a class of validators, and  
a specific portion that is specific to said operating system loader, and the



MSFT-3013/305832.01

replaceable with a different portion in the case where said validator is being used to validate a different operating system loader.

21. The system of claim 17, wherein said machine comprises a physical machine.

22. The system of claim 17, wherein said machine comprises a virtual machine.

23. A method of booting an operating system comprising:

executing a basic input output system, an option ROM, a master boot record, and a boot sector;  
starting an operating system loader;  
validating said operating system loader;  
validating a state of a machine on which said operating system loader executes;  
if said operating system loader, and said state of said machine, are determined to be valid, then allowing said operating system loader to load an operating system.

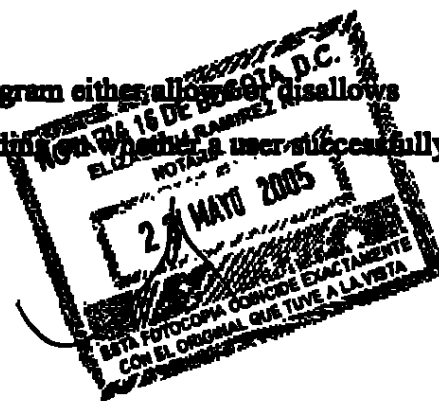
24. The method of claim 23, wherein a key that is necessary to correctly perform at least one function under said operating system is sealed to a validator that performs said acts of validating an operating system loader and validating a machine state, and wherein the method further comprises:

if the operating system loader and the machine state are valid, then unsealing said key and providing said key to said operating system loader.

25. The method of claim 23, further comprising:

after said operating system has been loaded, executing a logon program that gates access to said key.

26. The method of claim 23, wherein said logon program either allows or disallows components of said operating system to use said key depending on whether a user successfully completes an authentication procedure.



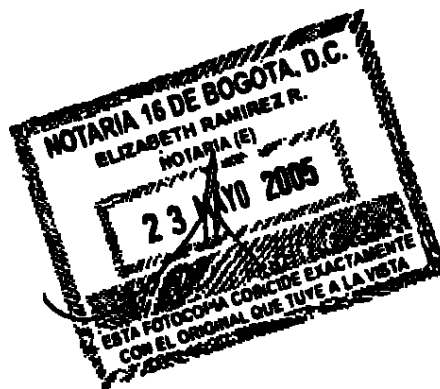
MSFT-3013/305832.01

27. The method of claim 23, wherein said acts of validating said operating system loader and validating said machine state are performed after said operating system loader has performed at least one action.

28. The method of claim 27, wherein said acts of validating said operating system loader and validating said machine state are performed at a point in time before resetting the machine state will prevent the loader from functioning correctly to load said operating system.

29. The method of claim 23, wherein said machine comprises a physical machine.

30. The method of claim 23, wherein said machine comprises a virtual machine.



**ABSTRACT OF THE DISCLOSURE**

A mechanism for protected operating system boot that prevents rogue components from being loaded with the operating system, and thus prevents divulgence of the system key under inappropriate circumstances. After a portion of the machine startup procedure has occurred, the operating system loader is run, the loader is validated, and a correct machine state is either verified to exist and/or created. Once the loader has been verified to be a legitimate loader, and the machine state under which it is running is verified to be correct, the loader's future behavior is known to protect against the loading of rogue components that could cause divulgence of the system key. With the loader's behavior being known to be safe for the system key, the validator may unseal the system key and provides it to the loader.



107

**PATENT APPLICATION  
DECLARATION AND POWER OF ATTORNEY**

ATTORNEY DOCKET NO. MFT-2004

MB DOCKET NO. 95532.01

I, a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention titled: **SYSTEM AND METHOD FOR PROTECTED OPERATING SYSTEM BOOT USING STATE VALIDATION**

A specification of which is filed herewith unless the following box is checked:

☐ was filed on \_\_\_\_\_ as US Application Serial No. or PCT International Application Number \_\_\_\_\_ and was amended on \_\_\_\_\_ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.54.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

| COUNTRY | APPLICATION NUMBER | DATE FILED | PRIORITY CLAIMED UNDER 35 U.S.C. 119 |
|---------|--------------------|------------|--------------------------------------|
|         |                    |            | YES NO                               |
|         |                    |            | YES NO                               |

**POWER OF ATTORNEY:**

As named inventor, I hereby appoint the following attorney(s) and/or agent(s) associated with

23377

|                                             |                                      |                                           |                                      |
|---------------------------------------------|--------------------------------------|-------------------------------------------|--------------------------------------|
| David Bartley Eppensauer<br>Reg. No. 31,499 | Martin L. Shively<br>Reg. No. 33,553 | Ronald O. Zink<br>Reg. No. 35,744         | Patrick E. Bowers<br>Reg. No. 37,038 |
| Ray Quan<br>Reg. No. 33,760                 | Jeffrey L. Rauck<br>Reg. No. 38,590  | Michael W. Bolesnowski<br>Reg. No. 28,692 | John Wersch<br>Reg. No. 32,332       |

I prosecute this application and conduct all business in the Patent and Trademark Office connected therewith.

**Lead Correspondence to:****Direct Telephone Calls To:**

Contact Name: Steven J. Reed  
Firm Name: Woodcock Washburn, LLP  
Firm Address: One Liberty Place, 46th Floor  
City, State and Zip: Philadelphia, PA 19103

Contact Name: Peter M. Ullman  
Contact Phone Number: 215-568-3180



## DECLARATION AND POWER OF ATTORNEY

ATTORNEY DOCKET NO. MSFT-3013

MS DOCKET NO. MSFT-3013

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of Inventor: Bryan Mark WillmanCitizenship: United States of AmericaAddress: Kirkland, WashingtonOffice Address: 10901 - 82nd Place NE, Kirkland, Washington 98034

Inventor's Signature

Date

Name of Inventor: Paul EnglandCitizenship: United KingdomAddress: Bellevue, WashingtonOffice Address: 14650 Northern Way, Bellevue, Washington 98008

Inventor's Signature

Date

6/29/04

Name of Inventor: Kenneth D. RayCitizenship: United States of AmericaAddress: Seattle, WashingtonOffice Address: 928 North 34th Street, 202, Seattle, Washington 98108

Inventor's Signature

Date

6/30/04

Name of Inventor: Ianina HunterCitizenship: United KingdomAddress: Bothell, WashingtonOffice Address: 16226 92nd Avenue NE, Bothell, Washington 98011

Inventor's Signature

Date

6/29/04

Name of Inventor: Leony Dawn McMichaelCitizenship: United States of AmericaAddress: Redmond, WashingtonOffice Address: 7980 23rd Ln NE, Redmond, Washington 98073

Inventor's Signature

Date

6-29-04



109

## DECLARATION AND POWER OF ATTORNEY

ATTORNEY DOCKET NO. MSPT-0011

MS DOCKET NO. MSPT-01

Full Name of Inventor: Derek Norman LaBelle

Citizenship: United States of America

Residence: Redmond, Washington

Post Office Address: 1802 162nd Ave. NE, Redmond, Washington 98052

Inventor's Signature

Date

Full Name of Inventor: Elmer J. Janssen

Citizenship: France

Residence: Sammamish, Washington

Post Office Address: 1221 20th Way, SE, Sammamish, Washington 98075

Inventor's Signature

Date

Full Name of Inventor: Mark Eliot Daly

Citizenship: United States of America

Residence: Redmond, Washington

Post Office Address: 18115 NE 51st Street, Redmond, WA 98052

Inventor's Signature

Date

6/29/04

Full Name of Inventor: Theodoraleksei Yevgeni Kuznetsov

Citizenship: United States of America

Residence: Sammamish, Washington

Post Office Address: 28130 NE 32nd Street, Sammamish, Washington 98074

Inventor's Signature

Date

6/29/04

Full Name of Inventor: David B. Cross

Citizenship: United States of America

Residence: Redmond, Washington

Post Office Address: 27024 NE 31st Court, Redmond, Washington 98053

Inventor's Signature

Date

6-29-04



\*\* TOTAL PAGE: 09 \*\*



2/5

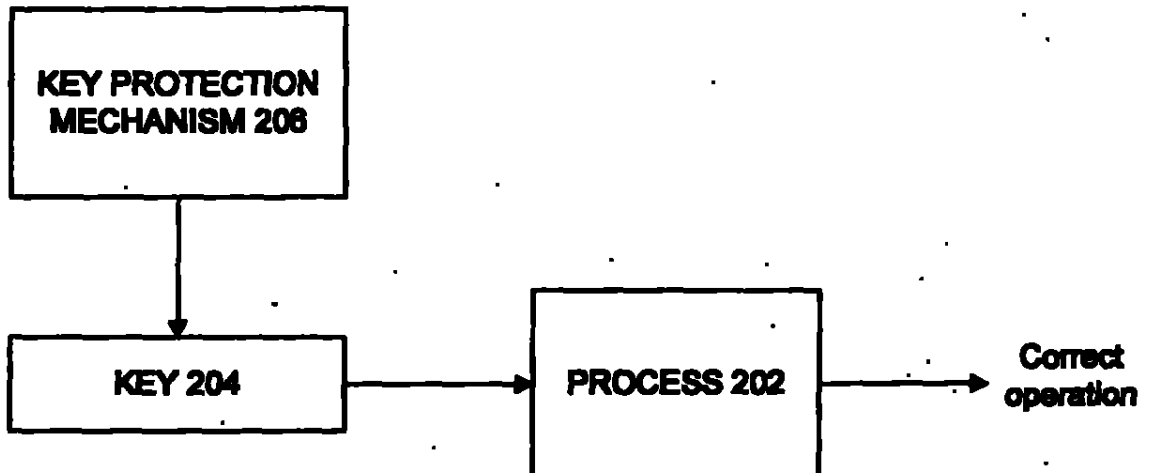


**FIG. 2**

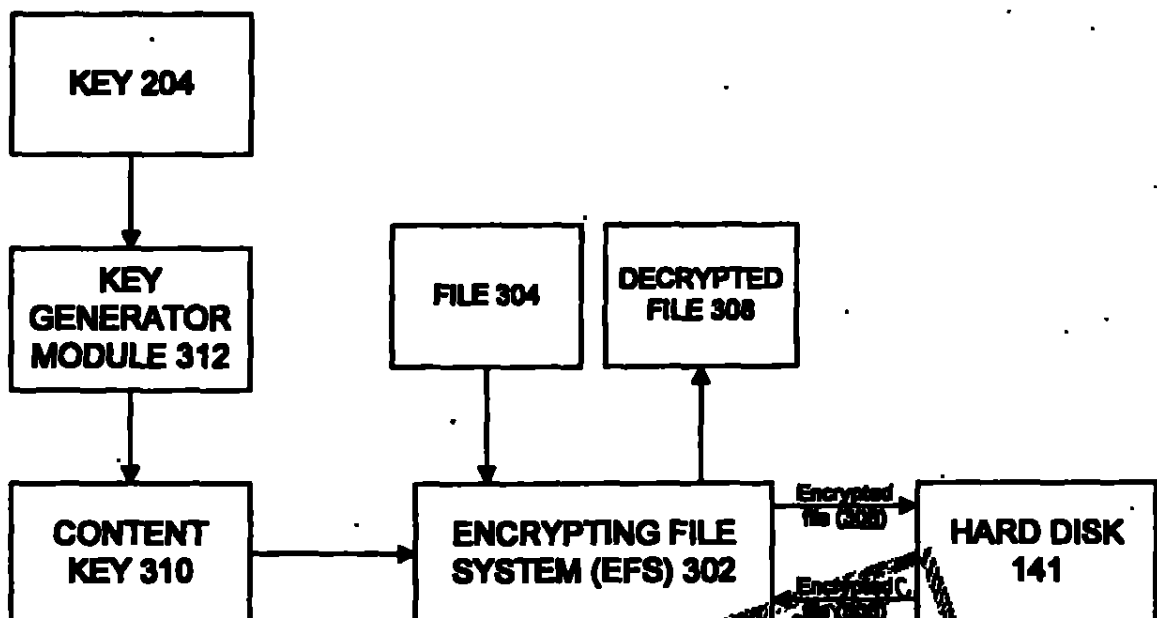
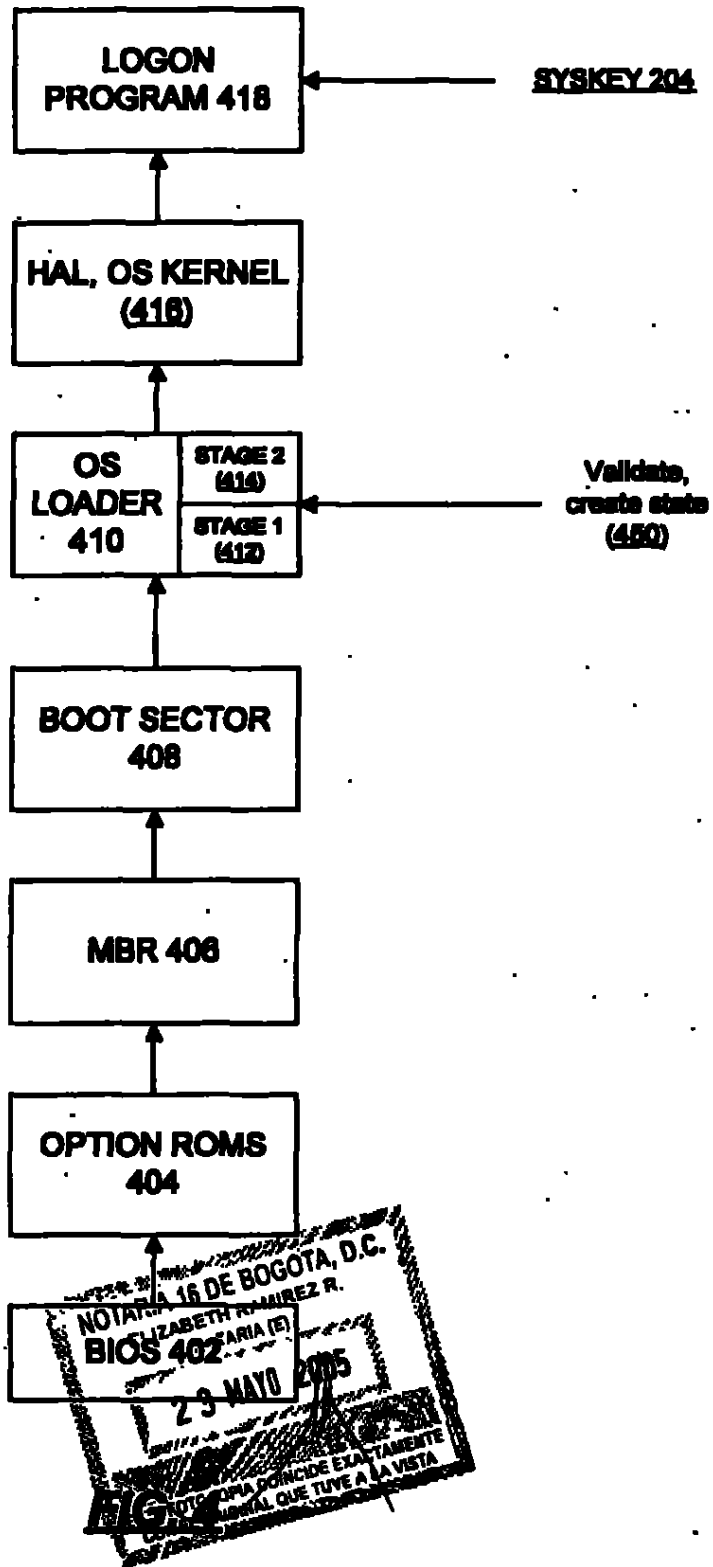


**FIG. 3**

3/5



4/5

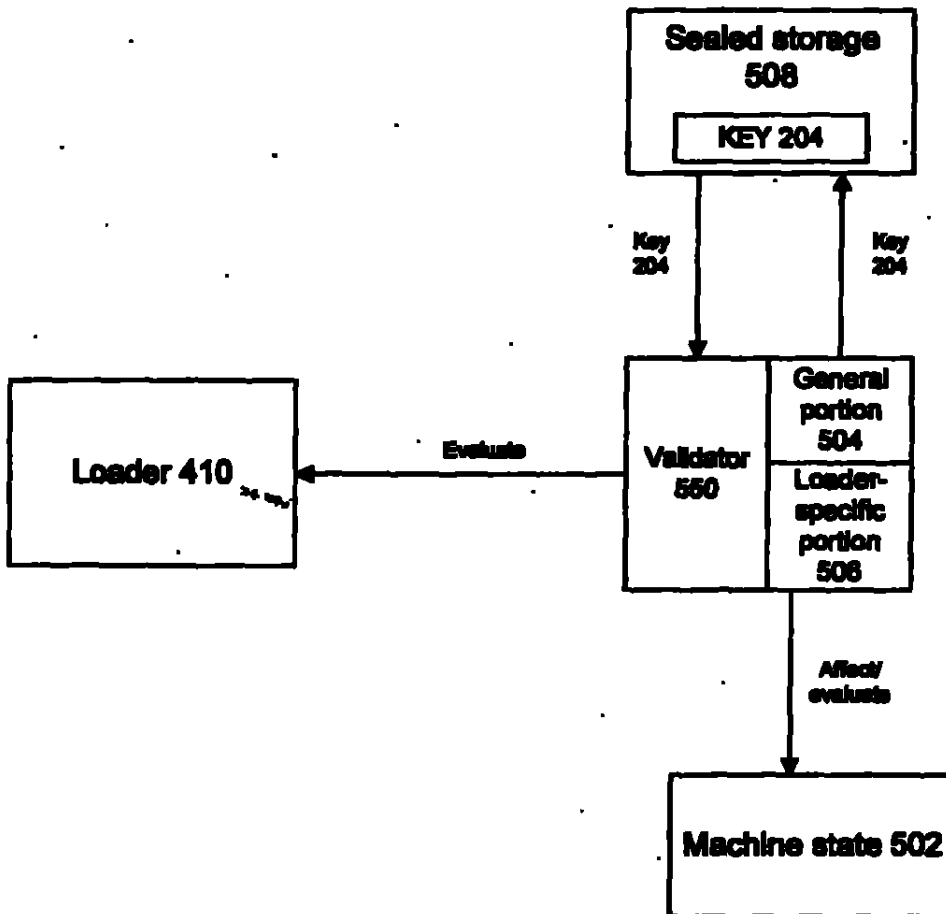
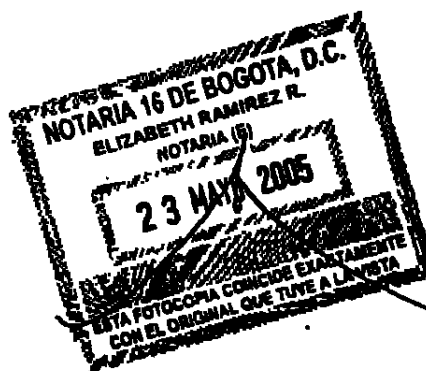


**FIG. 5**



114  
5/5

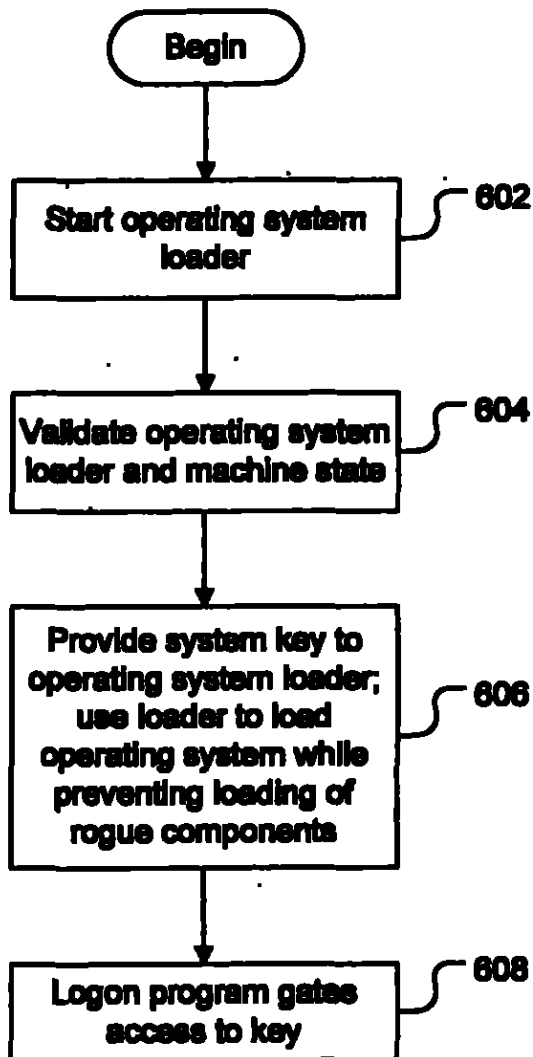
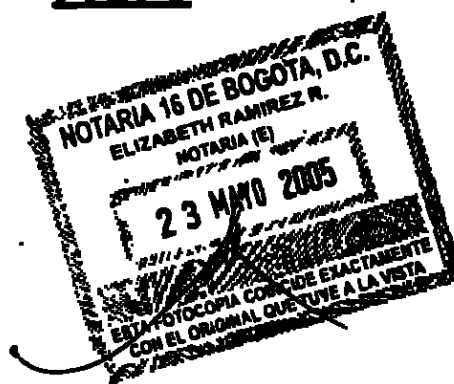


**FIG. 6**



Emitido en papelería especial con reborde negro-gris con el número PA 1295421 en la parte superior izquierda del documento. **"ESTADOS UNIDOS DE AMÉRICA - A TODOS LOS BENEFICIARIOS DE ESTE DOCUMENTO: EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS** Oficina de Patentes y Marcas Registradas de los Estados Unidos

16 de marzo de 2005

**ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS SON COPIA FIEL DE LOS DOCUMENTOS REGISTRADOS EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS ESTADOS UNIDOS CORRESPONDIENTES A LA SOLICITUD DE PATENTE IDENTIFICADA A CONTINUACIÓN, QUE CUMPLEN CON LOS REQUISITOS PARA QUE LES SEA OTORGADA UNA FECHA DE REGISTRO DE ACUERDO CON 35 USC 111.**

**NÚMERO DE SOLICITUD: 10/882,134**

**FECHA DE REGISTRO: 30 de junio de 2004**

Estampilla dorada del Departamento de Comercio - Oficina de Patentes y Marcas Registradas de los Estados Unidos

**Bajo la potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS**

(Firma)

**N. WOODSON**

Funcionario de Certificaciones

**SEGUNDA PÁGINA:** Registro del Abogado No. MSFT-3013/305832.01 PATENTE. SELLADO CON LOS NÚMEROS 22581 DEL U.S. PTO – 10/882134, CÓDIGO DE BARRAS 063004. A LA IZQUIERDA: SELLO CON EL NÚMERO 20427 U.S. PTO, CÓDIGO DE BARRAS 063004



**EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS  
ESTADOS UNIDOS**

En la solicitud de la referencia:

Bryan Mark Willman, Paul England

Kenneth D. Ray, Jamie Hunter, Lonny

McMichael, Derek LaSalle, Pierre Jacomet, Mark Paley, Thekkthalackal

Varugis Kurien, David B. Cross

Para: SISTEMA Y MÉTODO PARA INICIO DE SISTEMA OPERATIVO  
PROTEGIDO UTILIZANDO VALIDACIÓN DE LA CONDICIÓN

Etiqueta de Correo Expreso No. EL998514625US

Fecha de Depósito: junio 30 de 2004

**EL998514625US**

Comisionado de Patentes

P.O. Box 1450

Alexandria, VA 22313-1450

Señor:

**CARTA DE TRANSMISIÓN DE SOLICITUD DE PATENTE**

**Se emite por la presente lo siguiente para su registro:**

☒ Solicitud de Patente de Utilidad según 37 CFR 1.53 (b)

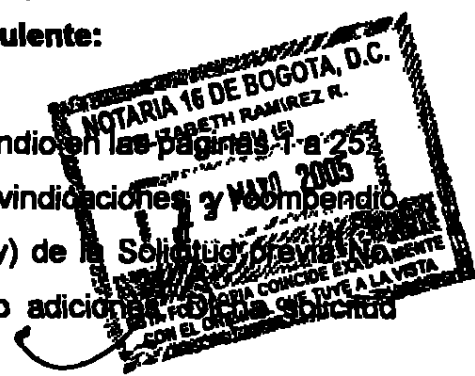
**La solicitud identificada previamente incluye lo siguiente:**

☐ Solicitud de no publicación

☒ Especificación que incluye reivindicaciones y compendio en las páginas 1 a 25

☐ Una copia de la especificación que incluye reivindicaciones y compendio  
(páginas \_\_ a \_\_) y las gráficas y anexos (si los hay) de la Solicitud de Patente

\_\_\_\_ presentada \_\_, al cual no se han realizado adicionales modificaciones  
previa queda incluida en esta solicitud por referencia.



Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005

**TERCERA PÁGINA:** Registro del Abogado No. MSFT-3013/305832.01 PATENTE.

☐ Una Nueva Declaración o Juramento sin Ejecutar y un Poder.

☒ Una Nueva Declaración o Juramento Ejecutada y un Poder ejecutado de Paul England, Kenneth D. Ray, Jamie Hunter, Lonny Dean McMichael, Mark Eliot Paley, Thekthalackal Varugis Kurien, David B. Cross (no se encuentra ejecutado para los otros 3 inventores)

☐ Se encuentra adjunta una copia de la declaración o juramento ejecutado presentado en la solicitud previa identificada anteriormente.

☐ Un Poder Asociado ejecutado por \_\_\_\_\_, quien se encuentra incluido en la copia de la Nueva Declaración o Juramento Ejecutada anexa.

☐ Una Copia del Poder con Revocatoria presentada en la Solicitud Previa No. \_\_\_\_\_.

☐ Declaración Firmada eliminando el (los) nombre (s) de los inventores incluidos en la solicitud previa.

☐ Esta solicitud reivindica prioridad en países extranjeros de acuerdo con 35 U.S.C. 119 de la Solicitud No. \_\_\_\_\_ presentado \_\_\_\_\_ en \_\_\_\_\_ (País).

☐ Una Copia Certificada de cada una de las solicitudes mencionadas arriba sobre las cuales se reivindica prioridad:

☐ Se encuentra anexa

☐ Fue presentada en la Solicitud Previa No. \_\_\_\_\_

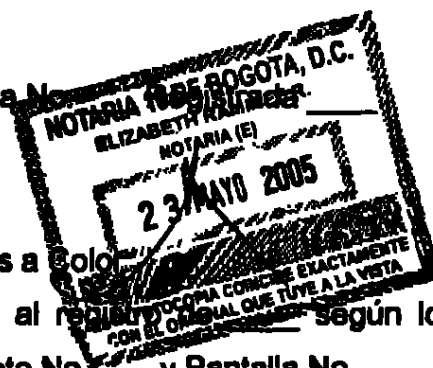
☒ 15 páginas de Gráficas Formales y/o fotografías.

☒ La Figura 4 deberá ser publicada.

☐ Se anexa solicitud para Aceptar Fotografías a Solo \_\_\_\_\_

☐ Esta solicitud de continuación/división se cede al registro de la solicitud de prioridad según lo incluido en la Cesión registrada el \_\_\_\_\_ en el Carrete No. \_\_\_\_\_ y Pantalla No. \_\_\_\_\_.

☐ Una Nueva Cesión Ejecutada que transfiere los derechos a Microsoft



Corporation.

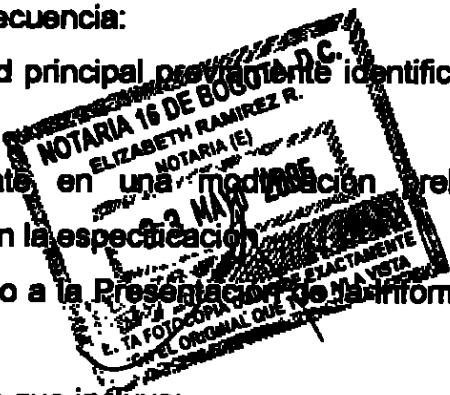
- ☐ Una Cubierta de Formato de Registro Legal
- ☐ Costo de Registro Legal - \$40.00
- ☐ Se anexa una Modificación Preliminar.
- ☐ Una Lista de Secuencia que consiste de páginas 1 - \_\_\_\_.

Página 2 de 5

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005

**CUARTA PÁGINA:** Registro del Abogado No. MSFT-3013/305832.01 PATENTE.

- ☐ Se anexa un diskette con la Lista de Secuencia.
- ☐ El formato legible en computador de la solicitud identificada previamente es idéntica a la presentada en la Solicitud No. \_\_\_\_, presentada \_\_\_\_\_. Según 37 CFR 1.821 (e), por favor utilice el listado de secuencia presentado el \_\_\_\_\_ y recibido por el PTO el \_\_\_\_\_ como el formato legible por el computador presentado en dicha solicitud como el formato legible por el computador presentado para esta solicitud. Se entiende que la Oficina de Patentes y Marcas Registradas hará las modificaciones en el número de solicitud y la fecha de presentación para el formato legible por el computador que será utilizado para esta solicitud.
- ☐ Una copia impresa del Listado de Secuencia:
  - ☐ fue presentada en la solicitud principal previamente identificada el \_\_\_\_\_ y recibida por PTO el \_\_\_\_\_.
  - ☐ se incluye por el presente en una modificación preliminar presentada por separado para su inclusión en la especificación.
- ☐ Se anexa una Declaración de Apoyo a la Presentación de Información de Secuencia.
- ☒ Declaración de Revelación de Información que incluye:



☒ Formato 1449.

☒ Copias de las Referencias 1-6 indicadas en el Formato Adjunto PTO-1449.

☐ Una Copia de la Solicitud para Extensión de Tiempo según lo presentado en el caso previo para co-dependencia

☒ Material Anexo según lo siguiente:

☒ Postal de Acuse de Recibo (debe estar indicado de forma específica).

Página 3 de 5

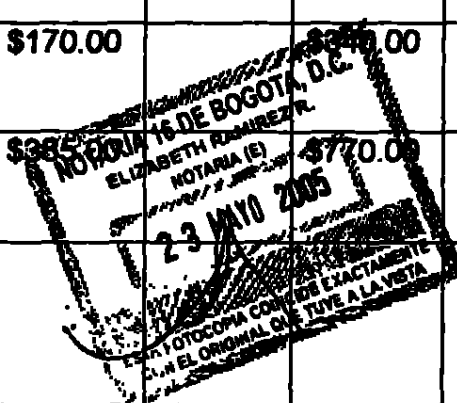
Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005

**QUINTA PÁGINA:** Registro del Abogado No. MSFT-3013/305832.01 PATENTE.

**CÁLCULO DE COSTOS:**

☐ El (los) solicitante (s) por medio de su abogado suscrito, reivindica su condición de entidad pequeña según 37 CFR 1.27 como \_\_\_\_ (ESCOGER UNO: Inventor Independiente, Pequeña Empresa o Entidad sin Ánimo de Lucro).

|                                                                           | PEQUEÑA ENTIDAD |       | NO PEQUEÑA ENTIDAD |          |
|---------------------------------------------------------------------------|-----------------|-------|--------------------|----------|
|                                                                           | TASA            | COSTO | TASA               | COSTO    |
| COSTO BASE DE SOLICITUD DE DISEÑO                                         | \$170.00        |       | \$340.00           |          |
| COSTO BASE DE SOLICITUD DE PATENTE DE UTILIDAD                            | \$385.00        |       | \$770.00           | \$770.00 |
| SOLICITUD DE PATENTE DE APLICACIÓN: TODAS LAS REIVINDICACIONES CALCULADAS |                 |       |                    |          |

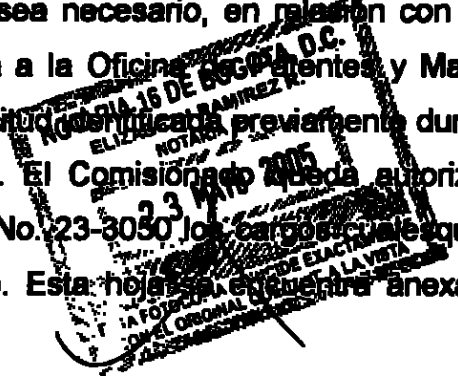


| DESPUÉS DEL INGRESO DE<br>TODOAS LAS MODIFICACIONES |                    |              |                     |  |                     |          |
|-----------------------------------------------------|--------------------|--------------|---------------------|--|---------------------|----------|
|                                                     | No.<br>Presentado. | No.<br>Extra |                     |  |                     |          |
| No. total de<br>reivindicaciones                    | 30 - 20 =          | 10           | \$9 cada<br>uno     |  | \$18<br>cada<br>uno | \$180.00 |
| No. total de<br>reivindicaciones<br>independientes  | 3 - 3 =            | 0            | \$43<br>cada<br>uno |  | \$86<br>cada<br>uno |          |
| COSTO DE REIVINDICACIÓN<br>DEPNIENTE MÚLTIPLE       |                    |              | \$145               |  | \$290               |          |
| COSTOS DE PRESENTACIÓN<br>ADICIONALES               |                    |              |                     |  |                     |          |
| COSTO TOTAL DE REGISTRO<br>DEBIDO                   |                    |              |                     |  |                     | \$950.00 |

X Se anexa un cheque por **\$950.00**. Por favor debite cualquier faltante o acredite cualquier sobrepago a la Cuenta de Depósito No. 23-3050.

   Por favor debite la Cuenta de Depósito No. 23-3050 por el monto de **\$0.00**. Esta hoja se encuentra anexa en duplicado.

X Se solicita por el presente al Comisionado conceder una extensión de tiempo por un período apropiado, en caso de que sea necesario, en relación con esta presentación o cualquier presentación futura a la Oficina de Patentes y Marcas Registradas de los Estados Unidos en la solicitud denunciada previamente durante el tiempo que esta solicitud esté pendiente. El Comisionado queda autorizado además para debitar la Cuenta de Depósito No. 23-3050 los cargos cualquiera relacionados con dicha extensión de tiempo. Esta hoja se encuentra anexa en duplicado.



X Toda Correspondencia deberá ser dirigida a Cliente No. 23377.

Código de Barras 23377 Oficina de Patentes y Marcas Registrada

Página 4 de 5

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005

**SEXTA PÁGINA:** Registro del Abogado No. MSFT-3013/305832.01 PATENTE.

X Todas las llamadas telefónicas deberán ser dirigidas a:

Peter M. Ullman

WOODCOCK WASHBURN LLP

One Liberty Place – 48th Floor

Philadelphia PA 19103

Teléfono: (215) 568-3100

Fax: (215) 568-3439

**EN CASO DE ENCONTRAR ALGUNA INCONFORMIDAD** con respecto a esta solicitud, incluyendo inconformidades en el pago de costos, partes faltantes de dicha solicitud u otros, se solicita respetuosamente a la Oficina de Patentes y Marcas Registradas de los Estados Unidos notificar inmediatamente al suscrito.

Fecha: 30 de junio de 2004

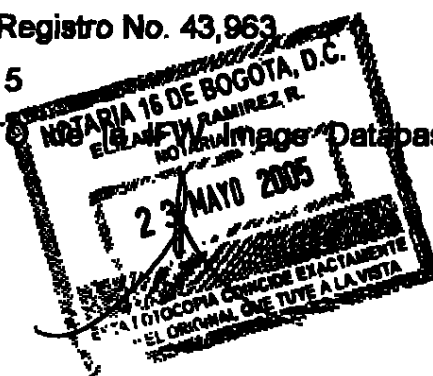
(Firma)

Peter M. Ullman

Registro No. 43,963

Página 5 de 5

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005



**MÉTODO Y SISTEMA PARA EL INICIO DEL SISTEMA OPERATIVO  
PROTEGIDO USANDO VALIDACIÓN DE ESTADO**

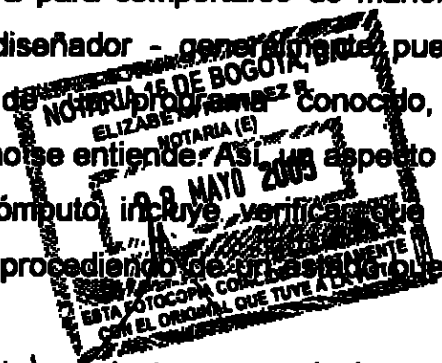
**CAMPO DE LA INVENCIÓN**

[0001] La invención presente generalmente se relaciona al campo de la computación. Más particularmente, la invención proporciona un mecanismo para asegurar que un sistema procede de un estado seguro conocido, y este mecanismo puede usarse para iniciar un sistema en cierto modo que proporciona seguridad suficiente del comportamiento correcto del sistema. Esta seguridad de comportamiento correcto, a su vez, puede impedir a uno o a más claves ser distribuidas bajo circunstancias inapropiadas.

**EL FONDO DE LA INVENCIÓN**

[0002] La seguridad de la computadora es a menudo dependiente de poder predecir el comportamiento de componentes de software. En general, la seguridad de un sistema puede fluir desde la premisa de que un programa conocido cuyo comportamiento es conocido, del cual procede de un estado bueno, actuará de una manera predecible. Recíprocamente, la frustración de seguridad - qué puede involucrar conseguir un sistema de computadora para comportarse de maneras que están fuera de la contemplación de su diseñador - generalmente puede comprenderse como reemplazo o cambio de un programa conocido, o ejecutándose en un estado en que su conducta no se entiende. Así, un aspecto de mantener la seguridad de un ambiente de cómputo incluye verificar que un programa conocido se está usando, y que está procediendo de un estado bueno conocido.

[0003] Un área dónde la predicción de conducta es particularmente importante está en la carga de un sistema operativo y sus componentes. Aunque



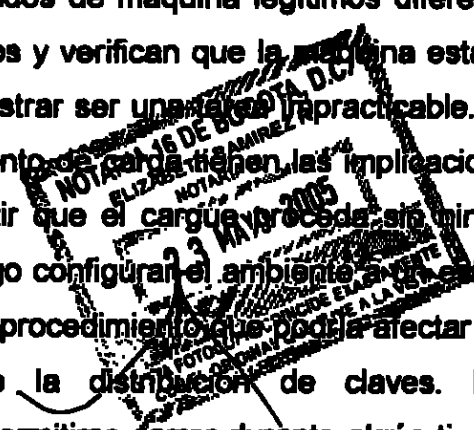
el propio sistema operativo puede diseñarse para proporcionar algún nivel de fidelidad acerca de su comportamiento, el tiempo antes de que un sistema operativo se ha cargado es un tiempo donde el sistema es particularmente vulnerable al ataque, desde la infraestructura que protege al sistema operativo de los ataques puede no haber sido establecido todavía (o puede estar en proceso de establecerse). Así, asegurando que el sistema operativo carga de una manera predecible es importante proteger el sistema operativo de ciertas clases de ataques.

[0004] Un tipo de brecha de seguridad que puede fluir desde un cargue no seguro (non-secure loading) de un sistema operativo relacionado a la protección de la clave (o claves) que habilita cierta funcionalidad restringida. Por vía del ejemplo pero sin limitación, el sistema operativo MICROSOFT WINDOWS emplean una clave del sistema, o "SYSKEY" que se usa para proteger varios procesos haciendo la actuación correcta de procesos dependientes de la disponibilidad de SYSKEY. Por ejemplo, la clave necesaria para descifrar la información privada que es guardada por el sistema operativo en forma cifrada puede ser los derivada del SYSKEY.

[0005] Convencionalmente, las claves necesitadas realizar las operaciones restringidas son protegidas por el procedimiento de autenticación (logon). Típicamente, el usuario debe autenticarse correctamente (por ejemplo, proporcionando las credenciales de autenticación correctas, como una combinación de usuario/clave (username/password) antes de comenzar el uso del sistema. El uso de las claves sólo se habilita si el usuario se autentica correctamente, y el sistema le permitirá un número limitado de intentos a un sólo usuario (por ejemplo, tres) antes de concluir que el usuario ha fallado la autenticación propiamente. (Este tipo de límite en el número de intentos al autenticarse les impide a los usuarios desautorizados habilitar uso de funcionalidades protegidas usando un ataque de fuerza bruto para adivinar la

contraseña en el caso de, digamos, una computadora portátil robada.) Sin embargo, usando el procedimiento de autenticación para proteger el acceso a las claves asume que cargar el sistema operativo correctamente cargado con el programa de autenticación correcto, y que el uso de las claves no se ha habilitado por otra parte por código malicioso que puede estar corriendo. Si un cargador malicioso fuera usado en cambio, y el cargador malintencionado causa una autenticación del programa maliciosa para ser cargados con el sistema operativo, entonces el uso de claves podría habilitarse, o las claves incluso podrían divulgarse, habiendo entrado sin las credenciales correctas. Desde que la carga del sistema operativo proporciona una brecha de oportunidad de seguridad, la protección de las claves en tal una situación requiere que la carga del sistema operativo tome lugar bajo circunstancias dónde puede verificarse que tome lugar correctamente.

[0006] Un problema que ocurre con verificar la seguridad de un proceso de carga de sistema operativo es que la cargue legítima del sistema operativo puede involucrar muchos programas diferentes (por ejemplo, hay numerosas "opciones diferentes de ROMs" que son pre-programas OS (sistema operativo) que corren durante un procedimiento de inicio de sistema), y hay numerosos procedimientos diferentes que pueden realizarse como parte de una carga de sistema operativo. Hay así, casi un incontable numero de estados de máquina legítimos diferentes durante una carga, e identifican tales estados y verifican que la máquina está en un estado bueno conocido que puede demostrar ser una tarea impracticable. Sin embargo, no todas las partes del procedimiento de carga tienen las implicaciones de seguridad. Puede ser más eficaz permitir que el cargue proceda sin ningún esfuerzo por evaluar su seguridad, pero luego configurar el ambiente al estado bueno conocido antes de empezar cualquier procedimiento que podría afectar una función relacionada de seguridad, como la distribución de claves. Más generalmente, un sistema arbitrario puede permitirse correr durante algún tiempo sin cualquier tipo de evaluación de seguridad, con tal de que el sistema pueda



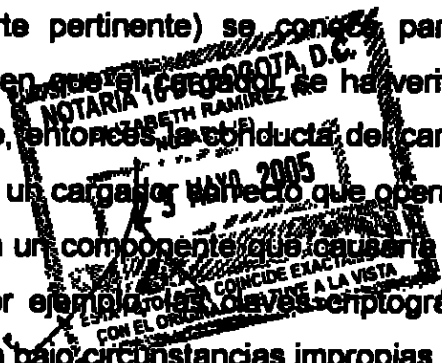
configurarse a un estado bueno conocido antes de permitir cualquier acción que puedan tener lugar implicaciones de seguridad.

[0007] En vista de lo anterior, hay una necesidad por un mecanismo que supere las desventajas del arte anterior.

**RESUMEN DE LA INVENCION**

[0008] La invención presente proporciona el cargue de un sistema operativo bajo circunstancias dónde la carga puede verificarse para ser realizada correctamente. Cuando una máquina se inicia, el procedimiento de arranque (startup) muy temprano (por ejemplo, BIOS, opciones de ROMs, registro de inicio master, sector de arranque, etc.) son realizados. Después de que estos procedimientos tempranos se realizan, el cargador del sistema operativo se inicia, y puede realizar varias tareas preliminares. Después de que el cargador del sistema operativo se inicia y ha realizado tales tareas preliminares, una aprobación del cargador del sistema operativo se ha realizado.

[0009] La aprobación comprende las pruebas de validez de representación en el propio cargador, o una porción del cargador (ej., checksums, u otras pruebas que se apuntan a evaluar la identidad y exactitud del cargador), así como evaluar el estado de la máquina actual (u obligando la máquina a conformar un estado bueno conocido). Si el cargador (o la parte pertinente) se conoce para ser correcto, y el estado de la máquina es uno en que el cargador se ha verificado para comportarse correctamente previamente, entonces la conducta del cargador puede predecirse. Así, puede asegurarse que un cargador correcto que opera bajo un estado de la máquina correcto no cargará un componente que causaría datos que habilitan las funciones protegidas o (por ejemplo, claves criptográficas, como la clave del sistema) para ser distribuida bajo circunstancias impropias.



**[0010]** Preferiblemente, la aprobación que se ha realizado por un validador que corre en un ambiente de alto-convicción. Un ambiente de convicción alto es aquel en que es posible proporcionar un grado relativamente alto de seguridad de que los procesos realizados en él correrán de la manera a que les esperan. Así, se asegura que el validador operará correctamente del hecho que el validador se verifica por un proceso que corre en el ambiente de alta-convicción, (por ejemplo, verificando una firma de su binario), y de la confianza subyacente que se procesa en el ambiente de alta-convicción que se llevará a cabo correctamente. (O, al menos hay un nivel de convicción con el cual el ambiente de alta-convicción no interferirá, o permite la interferencia correcta con la correcta operación de un proceso que opera dentro de tal medio ambiente, el funcionamiento correcto de un proceso que opera dentro de tal ambiente; uno todavía debe tener una base separada de confianza que la que el programa que lleva a cabo un proceso dentro del ambiente de alto-convicción realizará correctamente de la manera que se espera que realice.) Un ambiente de alta-seguridad puede proporcionar almacenamiento sellado, que es una facilidad del almacenamiento en que pueden sellarse los datos de un objeto particular (por ejemplo, a un programa particular), y que el control de ambiente de alta-seguridad es de tal manera que los datos sellados no se soltarán a cualquier objeto mas que el objeto al cual el dato es sellado (como es verificado por el ambiente de alta-seguridad). El validador puede usar este selló de almacenamiento para sellar una clave (por ejemplo, SYSKEY) a sí mismo, y puede negarse a abrir la clave excepto al objeto correcto, y cuando las circunstancias (por ejemplo, el estado de la máquina) es satisfactorio bajo algún estándar.

**[0011]** Otras características de la invención



**DESCRIPCIÓN BREVE DE LOS DIBUJOS**

[0012] El resumen anterior, así como la siguiente descripción detallada de personalizaciones preferidas, se entiende bien cuando es leída junto con los dibujos añadidos. Con el propósito de ilustrar la invención, allí se muestra en las construcciones de ejemplo de los dibujos de la invención; sin embargo, la invención no se limita a los métodos específicos y las utilidades expuestas. En los dibujos:

[0013] La FIG. 1 es un diagrama de bloque de un ejemplo de ambiente de cómputo en el cual pueden llevarse a cabo aspectos de la invención;

[0014] La FIG. 2 es un diagrama de bloque de un sistema que emplea un proceso cuyo funcionamiento correcto es dependiente de una clave del sistema;

[0015] La FIG. 3 es un diagrama de bloque de un sistema de archivo cifrado que protege los datos cifrados de ser descifrados sin autorización haciendo la descencripción dependiente de una clave del sistema;

[0016] La FIG. 4 es un diagrama de flujo de un proceso de arranque de ejemplo con la aprobación de acuerdo con los aspectos de la invención;

[0017] La FIG. 5 es un diagrama de bloque de un validador del ejemplo de acuerdo con los aspectos de la invención; y

[0018] La FIG. 6 es un diagrama de flujo de un proceso del ejemplo de proteger una clave del sistema de acuerdo con los aspectos de la invención;



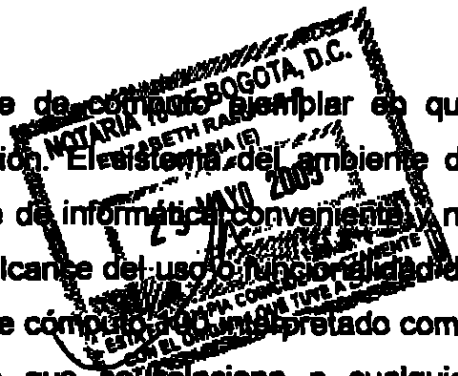
**DESCRIPCIÓN DETALLADA DE PERSONALIZACIONES ILUSTRATIVAS**

Apreciación global

**[0019]** Varios procesos que pueden tener lugar bajo un sistema operativo son dependientes de una o más claves para su operación correcto. El acceso a las claves puede ser controlado por un programa de autenticación, como un programa de logon que se niega a habilitar el uso de claves (key(s)) a menos que el usuario proporcione las credenciales correctas, como una combinación de usuario/clave. Así, por la negativa del programa de autenticación para habilitar el uso de claves en la ausencia de credenciales correctas, varios procesos (por ejemplo, descifrado de archivos cifrados) puede romperse (o completamente prevenirse) para usuarios que no conocen la contraseña. Mientras el programa de autenticación puede ser eficaz en el acceso de las claves, un cargador del sistema operativo podría engañarse en cargar un componente diferente que podría distribuir las claves fuera de las reglas de autenticación impuestas por el programa de autenticación. Así, cuando las claves son distribuidas de esta manera, mientras protege las claves requiere proteger el proceso cargante de sistema operativo. La invención presente proporciona mecanismos que pueden usarse para proteger el proceso cargante.

Arreglo de la Informática ejemplar

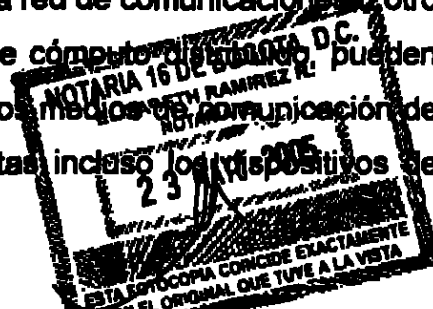
**[0020]** La FIG. 1 muestras un ambiente de cómputo ejemplar en que pueden llevarse a cabo aspectos de la invención. El sistema del ambiente de cómputo es sólo un ejemplo de un ambiente de informática conveniente, y no se piensa sugerir ninguna limitación acerca del alcance del uso o funcionalidad de la invención. Tampoco debería ser el ambiente de cómputo interpretado como para tener cualquier dependencia o requisito que se relaciona a cualquier



combinación de componentes ilustrados en el ambiente 100 de operación ejemplar.

**[0021]** La invención es operacional con numerosos otros propósitos generales o ambientes de sistema de cómputo o configuraciones especiales. Los ejemplos de sistemas de cómputo bien conocidos, ambientes, y/o configuraciones que pueden ser convenientes para el uso con la invención incluyen, pero no se limitan a, las computadoras personales, las computadoras del servidor, los dispositivos portátiles o portátiles, los sistemas de multiprocesador, los sistemas basados en microprocesador, las cajas de juego, la electrónica programable del consumidor, la red de PCs, miniordenadores, los sistemas informáticos grandes, los sistemas incluidos ("embebidos"), ambientes de cómputo distribuidos que incluyen cualquiera de los sistemas anteriores o dispositivos, y similares.

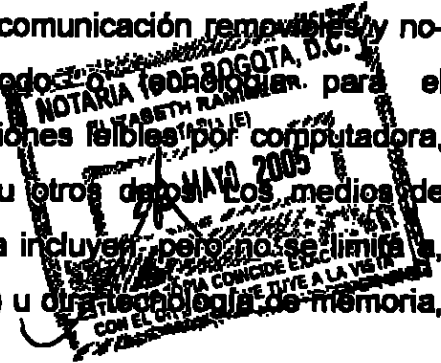
**[0022]** La invención puede describirse en el contexto general de instrucciones de computadora-ejecutables, como los módulos del programa, ejecutándose por una computadora. Generalmente, los módulos de programa incluyen las rutinas, programas, los objetos, los componentes, las estructuras de datos, etc. que realizan tareas particulares o implementan tipos de datos abstractos particulares. La invención también puede practicarse en ambientes de cómputo distribuidos donde las tareas son realizadas por dispositivos de procesamiento remotos que se unen a través de una red de comunicaciones u otro medio de transmisión de datos. En un ambiente de cómputo distribuido, pueden localizarse módulos de programa y otros datos en los medios de comunicación de almacenamiento de computadoras locales y remotas, incluso los dispositivos de almacenamiento de memoria.



**[0023]** Con referencia a la FIG. 1, un sistema ejemplar para llevar a cabo la invención incluye un dispositivo de cómputo de propósito general en la forma de una computadora 110. Los componentes de la computadora 110 pueden incluir,

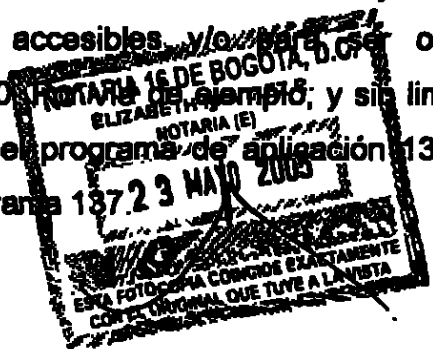
pero no se limitan a, una unidad de proceso 120, un sistema de memoria 130, y un sistema de bus 121 que se acopla a varios componentes del sistema incluida la memoria del sistema de la unidad de proceso 120. La unidad de proceso 120 puede representar unidades de proceso lógicas múltiples como aquéllos apoyados en unos procesadores multi-hilos. El sistema de bus 121 puede ser cualquiera de varios tipos de estructuras de bus incluso un bus de memoria o controlador de memoria, un bus periférico, y un bus local que usan cualquiera de una variedad de arquitecturas de bus. Por vía de ejemplo, y sin limitación, tales arquitecturas incluyen el bus de la Industria de la Arquitectura Estándar (ISA), el bus de Arquitectura de Micro Canal (MCA), el bus ISA Extendido (EISA), el autobús local de la Asociación de Estándares de la Electrónica de Video (VESA), y el bus de Componente Periférico de Interconexión (PCI) (también conocido como el bus del Entresuelo). El sistema de bus 121 también puede implementarse como una conexión punto a punto, cambio de manufactura, o similares, entre los dispositivos de comunicación.

[0024] La computadora 110 típicamente incluye una variedad de medios legibles por computadora. Los medios legibles por computadora pueden ser cualquier medio de comunicación disponible que puede accederse por una computadora 110 y pueden ser incluidos ambos medios de comunicación volátil y no volátil, los medios de comunicación removibles y no-removibles. Por vía de ejemplo, y sin limitación, los medios legibles por computadora pueden comprender medios de comunicación de almacenamiento de computadora y medios de comunicación. Los medios de comunicación de almacenamiento de computadora incluyen ambos volátil y no volátil, los medios de comunicación removibles y no-removibles implementados en cualquier método para el almacenamiento de información como las instrucciones legibles por computadora, las estructuras de datos, módulos de programa u otros datos. Los medios de comunicación de almacenamiento de computadora incluyen, pero no se limita a, RAM, ROM, EEPROM, memoria de destello (flash) u otra tecnología de memoria,



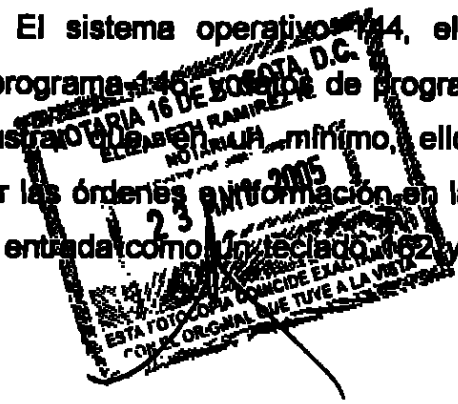
CDROM, los discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnetofónica, almacenamiento del disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que puede usarse para guardar la información deseada y que es accedida por la computadora 110. Los medios de comunicación incluyen típicamente las instrucciones leíbles por computadora, las estructuras de datos, los módulos de programa u otros datos en señales de datos moduladas como una onda portadora u otro mecanismo de transporte e incluyen cualquier medio de comunicación de entrega de información. El término "señal de datos modulada" es una señal que tiene una o más de un conjunto de características o cambia de tal manera para codificar la información en la señal. Por vía del ejemplo, y sin limitación, los medios de comunicación incluyen los medios de comunicación alambrados como una red alambrada o conexión directa-alambraron, y los medios de comunicación inalámbricos como el acústico, RF, infrarrojo y otros medios de comunicación inalámbricos. Las combinaciones de cualquiera de los anteriores también deben ser incluido dentro del alcance de los medios de comunicación leíbles por computadora.

[0025] El sistema de memoria 130 incluye los medios de comunicación de almacenamiento de computadora en forma volátil y/o memoria no volátil como la memoria de sólo lectura (ROM) 131 y la memoria de acceso aleatorio (RAM) 132. Un sistema básico de entrada/salida (input/output) (BIOS) 133, conteniendo las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 110, como durante el arranque (Start-up), que se guarda típicamente en ROM 131. La RAM 132 típicamente contiene los datos y/o módulos del programa que son inmediatamente accesibles y/o para ser operados presentemente por la unidad de proceso 120. Por vía del ejemplo, y sin limitación, la Fig. 1 ilustra el sistema operativo 134, el programa de aplicación 135, otros módulos de programas 136, y datos de programa 137.



[0026] La computadora 110 también puede incluir otros medios de almacenamiento de computadora removible/no- removible, volátil/no- volátil. Sólo, por vía de ejemplo la Fig. 1 ilustra una unidad de disco duro 141 que lee desde o escribe sobre los medios magnéticos no-removibles, no-volátiles, una unidad de disco magnético 151 que lee desde o escribe en un disco magnético 152 removible, no-volátil, y una unidad de disco óptico 155 que lee desde o escribe en un disco óptico 156 removible, no-volátil, como un CD ROM u otro medio óptico. Otros medios de almacenamiento de computadora removible/no- removible, volátil/no- volátil que pueden usarse en el ambiente de operación ejemplar incluyen, pero no se limitan a, los casetes de cinta magnetofónica, la tarjetas de memoria de destello (flash), discos versátiles digitales, cinta de video digital, RAM de estado sólida, ROM de estado sólido, y similares. La unidad de disco duro 141 se conecta típicamente al sistema de bus 121 a través de una memoria no-removible como una interfaz 140, y se conectan a la unidad de disco magnético 151 y la unidad de disco óptico 155 típicamente al sistema de bus 121 por una memoria removible, como la interfaz 150.

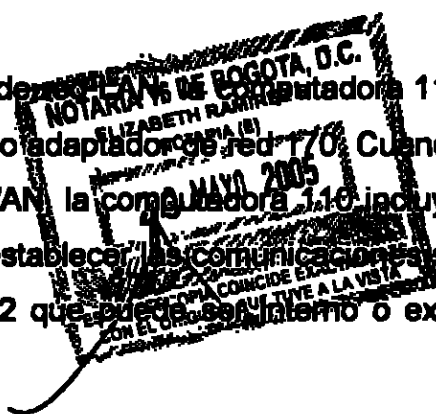
[0027] Las unidades y sus medios de almacenamiento de computadora asociados discutidos arriba e ilustrados en la Fig. 1, proporcionan almacenamiento de instrucciones leíbles por computadora, las estructuras de datos, los módulos de programa y otros datos para la computadora 110. En la Fig. 1, por ejemplo, la unidad de disco duro 141 se ilustra como almacén del sistema operativo 144, del programa de aplicación 145, de otros módulos de programa 146, y datos de programa 147. Note igual que estos componentes pueden ser diferentes del sistema operativo 134, el programa de aplicación 135, otros módulos de programa 136, y datos de programa 137. El sistema operativo 144, el programa de aplicación 145, otros módulos de programa 146 y datos de programa 147 tienen aquí números diferentes para ilustrar que son copias diferentes. Un usuario puede entrar las órdenes de información en la computadora 20 a través de los dispositivos de entrada como un teclado 162 y dispositivo de



apuntamiento 161, normalmente llamado ratón, bola rotativa (trackball) o almohadilla de contacto. Otros dispositivos de entrada (no mostrados) puede incluir un micrófono, una palanca de mando, la almohadilla de juego, un plato satélite, escáner, o similares. Se conectan a menudo éstos y otros dispositivos de entrada a la unidad de proceso 120 a través de una interfaz de usuario 160 que se acopla al sistema de bus, pero puede conectarse por otra interfaz y estructuras de bus, como un puerto paralelo, un puerto de juego o un bus de serie universal (USB). Un monitor 191 u otro tipo de dispositivo de despliegue también se conecta al sistema de bus 121 vía una interfaz, como una interfaz de video 190. Además del monitor, las computadoras pueden incluir también otros dispositivos de rendimiento periféricos como parlantes 197 e impresora 196 que pueden conectarse a través de una interfaz periférica de salida 195.

[0028] La computadora 110 puede operar en un ambiente conectado a una red de computadoras que usa las conexiones lógicas a uno o más computadoras remotas, como una computadora remota 180. La computadora remota 180 puede ser una computadora personal, un servidor, un enrutador, una PC de red, un dispositivo par u otro nodo de la red común, y típicamente incluye muchos o todos los elementos descritos arriba relacionados a la computadora 110, aunque sólo un dispositivo de almacenamiento de memoria 181 se ha ilustrado en la Fig. 1. Las conexiones lógicas ilustradas en la Fig. 1 incluyen una red de área local (LAN) 171 y una red de área ancha (WAN) 173, pero también puede incluir otras redes. Tales ambientes de gestión de redes son comunes en las oficinas, en redes de computadoras de empresas grandes, intranets y la Internet.

[0029] Cuando se usa en un ambiente de red LAN, la computadora 110 se conecta a la LAN 171 a través de una interfaz o adaptador de red 170. Cuando es usado en un ambiente de gestión de redes WAN, la computadora 110 incluye un módem 172 u otros medios típicamente para establecer las comunicaciones sobre la WAN 173, como la Internet. El módem 172 que puede ser interno o externo

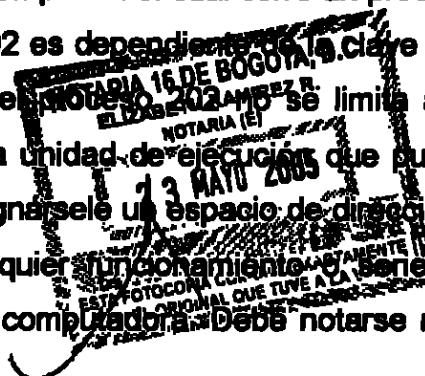


puede conectarse al sistema de bus 121 vía la interfaz de usuario 160, u otro mecanismo apropiado. En un ambiente conectado una red de computadoras, los módulos de programa representados relacionados a la computadora 110, o parte de eso, puede guardarse en el dispositivo de almacenamiento de memoria remoto. Por vía del ejemplo, y sin limitación, la Fig. 1 ilustra los programas de aplicación remota 185 como residentes en el dispositivo de memoria 181. Se apreciará que las conexiones de red mostradas son medios ejemplares y otros medios de establecer vínculos de comunicación entre las computadoras pueden ser usados.

### Acción protegida por las claves

[0030] Un ambiente de cómputo puede emplear una clave sobre la cual ciertos procesos que tienen lugar dentro del ambiente dependen para su funcionamiento correcto. La clave del sistema, o "SYSKEY" usado por los sistemas operativos MICROSOFT WINDOWS es un ejemplo de tal clave, pero no es un ejemplo limitando. En una personalización preferida, una clave sobre la que el proceso depende es única, criptográficamente clave aleatoria por-plataforma - es decir, dadas dos máquinas, es probable que las dos máquinas tengan claves diferentes. Así, un proceso que es dependiente de tales claves es improbable que sea portátil de una plataforma a otro - por lo menos en la magnitud que se emplean los mecanismos eficaces para asegurar que la clave de una plataforma no está disponible fuera de esa plataforma.

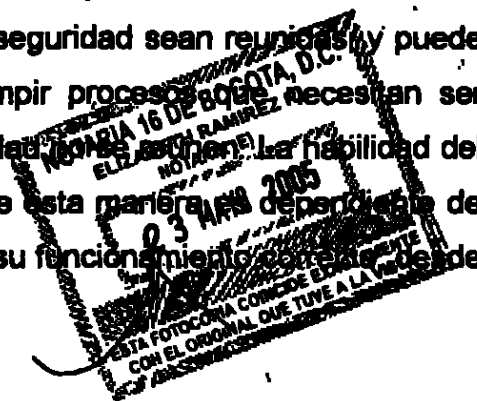
[0031] La Fig. 2 muestra un sistema de ejemplo en el cual corre un proceso que es dependiente de una clave. El proceso 202 es dependiente de la clave 204 para operar correctamente. Debe notarse que el proceso 202 no se limita a la noción tradicional de un proceso - es decir, una unidad de ejecución que puede manejarse por un sistema operativo y puede asignarse un espacio de dirección - sino que generalmente se refiere más a cualquier funcionamiento o serie de funcionamientos que pueden realizarse en una computadora. Debe notarse más



allá que, mientras este ejemplo muestra un proceso que es dependiente de una clave criptográfica, el término proceso como es usado aquí dentro no se limita a procesos que realizan las operaciones criptográficas.

[0032] Como es mostrado en la Fig. 2, si la clave 204 está disponible como una entrada de proceso 202, entonces el proceso 202 opera correctamente. Por otro lado, si la clave 204 no está disponible como una entrada de proceso 202, entonces el proceso 202 no opera correctamente. El mecanismo de protección 206 controla el acceso a la clave 204 - es decir, el mecanismo 206 proporciona cualquiera, o no proporciona, el proceso 202 con la clave 204, dependiendo de si las condiciones de seguridad pertinentes se han reunido. Por ejemplo, un usuario puede tener que autenticarse y proporcionar una contraseña correcta antes de que el mecanismo 206 habilite el uso de la clave 204.

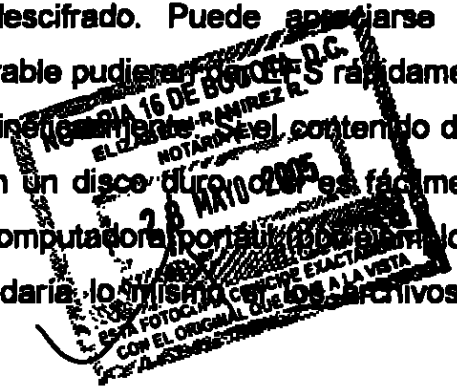
[0033] Debe notarse que el rechazo de clave 204 para impedir que el proceso 202 opere correctamente a veces es un resultado deseado. Por ejemplo, el cifrado/descifrado (encryption/decryption) de archivos es un ejemplo de un proceso 202 que es protegido por la clave 204. El descifrar correctamente un archivo puede depender del acceso a la clave 204. Si el usuario no puede correctamente autenticarse, entonces puede ser deseable para el descifrado de un archivo no proceder, desde la incapacidad de un usuario para autenticarse puede indicar que la computadora está operándose por alguien diferente de su usuario intencional (por ejemplo, en el caso de una computadora portátil robada). Así, la clave de protección del mecanismo 206 puede hacer que el acceso a la clave 204 dependa de que las condiciones relevantes de seguridad sean reunidas, y puede usar el rechazo de la clave 204 para interrumpir procesos que necesitan ser interrumpidos cuando las condiciones de seguridad no se reúnen. La habilidad del mecanismo 206 para interrumpir los procesos de esta manera es dependiente de la clave 204 requiriendo de esos procesos para su funcionamiento.



que esta dependencia habilita el acto de negar la clave 204 para interrumpir el proceso.

[0034] La Fig. 3 muestra un específico (pero no limitante) ejemplo de un proceso que es dependiente de la clave 204. En el ejemplo de la Fig. 3, el proceso del ejemplo es un Sistema de Archivos Encriptado (EFS) 302 que guarda los archivos en forma cifrada, y también descifra los archivos cifrados. Debe notarse que un propósito de tal encriptación de archivo es para proteger los datos en una computadora portátil de recuperarse por un ladrón en caso de que la computadora portátil sea robada. Cuando un archivo 304 se genera para el almacenamiento, el archivo se proporciona para EFS 302. EFS 302 cifra el archivo 304, y convierte el archivo 304 en un archivo 306 cifrado que se guarda en disco duro 141. Cuando una demanda se hace para recuperar el archivo 306 cifrado, EFS 302 recupera el archivo 306 cifrado y lo descifra para generar el archivo 308 descifrado. (En la práctica, el meollo del archivo 308 descifrado es igual que el archivo 304 original, aunque para claridad la Fig. 3 muestra estos dos casos de archivo separadamente: el archivo 304 es el archivo original, y el archivo descifrado 308 es el mismo archivo después de que se ha cifrado, guardado, recuperado, y descifrado por EFS 302.)

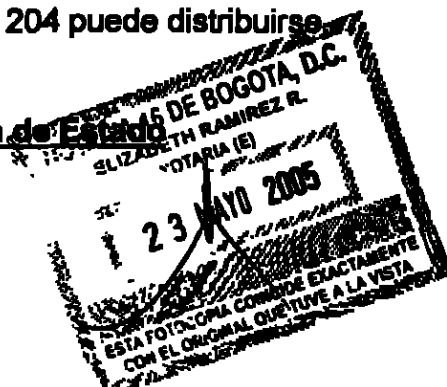
[0035] Se notará que una entrada a EFS 302 es la clave 310 satisfecha. La clave 310 satisfecha es preferiblemente una clave simétrica que funciona como una entrada a un método criptográfico. La clave 310 satisfecha se usa para cifrar el archivo 304 y crear el archivo 306 cifrado, y se usa para descifrar el archivo 306 también cifrado y crear el archivo 308 descifrado. Puede apreciarse que guardando la clave 310 en algún lugar recuperable pudieran ser EFS rápidamente a la habilidad de 302 para proteger los datos instantáneamente. El contenido de la clave satisfecha está fácilmente disponible en un disco duro 141 es fácilmente determinable de algún rasgo conocido de la computadora portátil (por ejemplo, el número de serie del procesador), entonces daría lo mismo. Los archivos se



guardaron en forma cifrada, desde que un ladrón pudiera encontrar la clave fácilmente y podría descifrarlos. Así, es deseable proteger la clave derivándolo de alguna manera que sólo puede hacerse con la cooperación del verdadero dueño. Una manera de proteger la clave es usar un generador importante módulo 312 que recibe la clave 204 como la entrada y deriva la clave 310 satisfecha como una función de la clave 204. Así, para extender que la clave 204 se proporcionará solo bajo las circunstancias apropiadamente seguras, es igualmente verdad que la clave 310 satisfecha será sólo derivable bajo las circunstancias apropiadas. En otros términos, haciendo la derivación de clave 310 satisfecha dependiente de la disponibilidad de la clave 204, cualquier protección que se permite para codificar la clave 204 puede extenderse al contenido de la clave 310. Por ejemplo, si proporcionar la clave 204 requiere que el usuario se autentique proporcionando una contraseña correcta, puede asegurarse que el contenido de la clave 310 no estará disponible a menos que el usuario se autentique correctamente.

[0036] Así, la protección de la clave 204 de ser dada bajo circunstancias erróneas es importante, desde que otros procesos pueden depender de la clave 204 que sólo se proporciona cuando el contexto de seguridad es correcto (por ejemplo, una legítimo, usuario autenticado que proporciona la contraseña correcta) está presente. Como es descrito abajo, una manera de causar que una máquina distribuya la clave 204 de una forma que puede producir mal uso es iniciar la máquina en un modo no seguro en el cual los componentes falsos del sistema operativo pueden reemplazar los componentes correctos (donde los componentes correctos se presumen para poder proteger la clave 204). Así, se describen los mecanismos debajo que aseguran que una máquina es iniciada en un ambiente conocido (seguro) antes de que la clave 204 puede distribuirse.

#### El Proceso de Inicio con la Validación de Estado

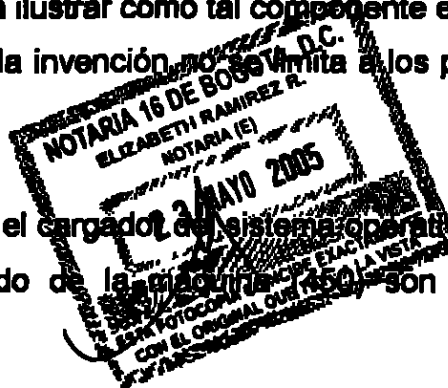


**[0037]** La Fig. 4 muestra una sucesión típica de eventos que se usan en un procedimiento de inicio (booteo).

**[0038]** Inicialmente, la máquina es encendida. Una máquina típica se configura para empezar a ejecutar instrucciones en alguna dirección fija en el momento que es encendida. Las instrucciones típicamente contenidas en esta dirección son conocidas como "BIOS" 402, o "Sistema de entrada/salida (Input/Output) Básico". En la conclusión de ejecución de la BIOS 402, La BIOS 402 inicia la ejecución de pequeños programas llamados "opciones de ROM" 404. Las opciones de ROM son programas que realizan las funciones básicas de arranque temprano, tal como configurar la contraseña de hardware para la máquina, o seleccionar cuál de varios sistemas operativos será iniciado. Después de que la opción ROM 404 se ejecuta, la máquina es instruida para cargar el Registro Principal de Arranque (MBR – Master Boot Record) 406.

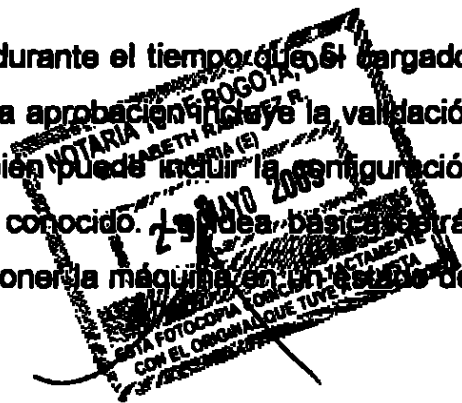
**[0039]** El MBR 406 es un programa ejecutable. Típicamente, el MBR 406 reside en el primer sector del disco duro de una computadora, y empieza por buscar en la tabla de particiones que partición debe usar para el arranque. (Por ejemplo, un disco puede dividirse para el uso con sistemas operativos diferentes, y cada sistema operativo puede requerir un procedimiento de arranque diferente.) Después de que la partición correcta se ha buscado, el MBR 406 transfiere el control al sector de arranque 408 asociado con esa partición. El sector de arranque entonces comienza el proceso de cargar el sistema operativo cargador 410 que cargará el sistema operativo finalmente. Debe notarse que el MBR 406 se muestra meramente en la Fig. 4 para ilustrar cómo tal componente encajaría en un proceso de arranque de ejemplo, y la invención no se limita a los procedimientos de arranque que usan MBR 406.

**[0040]** Durante el tiempo que el cargador de sistema operativo 410 se está ejecutando, el cargador y el estado de la máquina son validados. (la



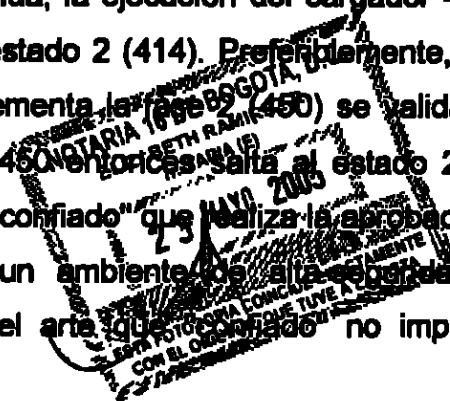
Máquina, en este contexto, puede referirse a una máquina física o una máquina virtual.) La validación se ha realizado por un componente de software confiable que opera en un ambiente de alta-seguridad (una personalización de cual es más particularmente descrita debajo), y por lo tanto hay algún grado de seguro/confiabilidad (assurance/trustworthiness) que la validación del cargador 410 es hecho correctamente. En esencia, para validar los aspectos pertinentes del estado de la máquina mientras el cargador del sistema operativo 410 tiene el control de la máquina, y valida el cargador 410, es posible proporcionar algún nivel de seguridad que ilegítima o falsea las versiones de los componentes del sistema operativo (por ejemplo, la capa de abstracción de hardware, el núcleo (kernel), los controladores, etc. (416)) no se cargará después, al tiempo que la autenticación del programan 418 es ejecutado. Prevenir que componentes falsos sean cargados al tiempo que la autenticación del programan 418 es ejecutada es importante porque uno de las primeras cosas que el sistema operativo hará después de que está cargado es ejecutar un programa de autenticación 418 que conduce el acceso de la clave 204, y si pudieran cargarse los componentes falsos, entonces estos componentes podrían causar que el programan 418 se comporte mal de una manera que podría producir distribución de la clave 204 bajo circunstancias impropias, por lo tanto comprometiendo la seguridad de todos los componentes de los que dependen de la protección de clave 204 como fue descrito anteriormente. Así, la protección de la clave 204 puede lograrse ejerciendo el control correcto sobre el estado de la máquina del tiempo desde que cargador del sistema operativo 410 se ejecuta hasta que la autenticación del programa se ha completado.

[0041] La validación 450 toma lugar durante el tiempo que el cargador del sistema operativo 410 se está ejecutando. La aprobación incluye la validación del cargador y el estado de la máquina, y también puede incluir la configuración del estado de la máquina a un estado bueno conocido. La idea básica detrás de validar y poner el estado de la máquina es poner la máquina en un estado donde,



si el cargador se ejecuta mientras la máquina está en ese estado, entonces el cargador no cargará ningún componente falso o por otra parte se comportará de manera que producirían la brecha de seguridad. La validación 450 asegura que el código del cargador es, de hecho, el código que se ha verificado para comportarse correctamente previamente, y también asegura que la máquina está en un estado bajo el cual el código es conocido se comportará correctamente (verificando que la máquina ya está en ese estado, o poniendo la máquina en ese estado). Se apreciará que esta técnica puede apalancar el existente estado de seguridad de la máquina para asegurar que los estados futuros de la máquina también estarán seguros, y este apalancamiento es hecho posible por el hecho que el comportamiento del cargador del sistema operativo 410 no sólo es conocido y entendido, pero también herméticamente circunscrito. Basado en esta observación, se apreciará que la validación 450 no debe tomar demasiado temprano (por ejemplo, en el momento que la opción de ROM 404, o MBR 406, se ejecuta), desde la variedad amplia de código de las numerosas fuentes diferentes, y la inmensa anchura de estados que la máquina podría ponerse como resultado de ejecutar tal código, no lo hace difícil, si no imposible, para determinar el comportamiento de la máquina mientras se ejecutan todos estos varios módulos de código diferentes. Así, es preferible no tener relación con qué los estados de la máquina sean previos a la ejecución del cargador 410, con tal de que la máquina pueda ponerse en un estado legítimo en el momento que el cargador 410 se está corriendo.

[0042] En una personalización preferida, la ejecución del cargador 410 es dividida en dos estados: estado 1 (412) y estado 2 (414). Preferiblemente, antes de que la fase 2 entre, el código que implementa la fase 2 (450) se valida, y el código confiado que realiza la aprobación 450 entonces pasa al estado 2 a un punto de entrada bien-definido. (El "código confiado" que realiza la aprobación es el programa antes dicho que corre en un ambiente de alta seguridad. Se entenderá por aquéllos con habilidad en el arte que el código confiado no implica la

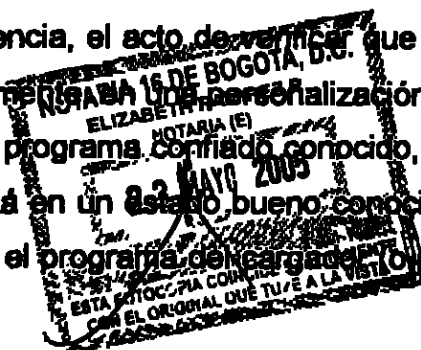


infalibilidad absoluta, pero meramente significa que hay alguna base para asumir que el código hará su trabajo correctamente. Desde que el comportamiento de un programa puede afectarse por el ambiente en que corre, mientras se ejecuta el código confiado en un medio ambiente de alta-seguridad el código confiado operará correctamente: (1) el código confiado puede confiarse para realizar su función correctamente en un ambiente que conforma algunas expectativas, y (2) en el ambiente de alta-seguridad puede confiarse para proporcionar un ambiente que conforma tales expectativas correctamente.) El estado 2 puede validar entonces (según algunos la norma definida) cualquier información que ha recibido el estado 1. La línea divisoria entre el estado 1 y el estado 2 refleja el hecho de que puede haber algunos aspectos de un procedimiento de arranque que pueden realizarse sin ninguna comprobación de seguridad, y sin tener ninguna consecuencia para alguna tarea de seguridad-relacionada (como la distribución de la clave 204), con tal de que puedan validarse los programas importantes y el estado en algún punto antes de que pudieran ponerse los eventos en movimiento que podría causar que una tarea de seguridad-relacionada no sea realizada correctamente. Más aun, puede haber problemas, con intentar validar el estado demasiado temprano en el proceso de arranque: la ruta de ejecución real que legitima el proceso de arranque que puede tomar es bastante inconstante, por eso es difícil definir la diferencia entre los estados de la máquina válidos o no válido bajo tales circunstancias inconstantes. Puede tener sentido para permitir la carga simplemente proceder a través de cualquiera de estos estados inconstantes, sin intentar determinar si algún estado a través del cual la máquina pasa es válido. Así el estado 1 representa la porción de la carga que procede sin ninguna aprobación. Al final del estado 1, se validan el cargador y el estado de la máquina, y el estado 2 empieza. La presunción de una línea divisoria es que los procesos pueden ejecutarse sin ninguna restricción de seguridad. NOTARIA 16 DE BOGOTÁ, D.C. ELIZABETH R. NOTARIA (E) 31 MAYO 2003 ESTÁ FOTOCOPIA CANCIONAMENTE CON EL ORIGINAL QUE TUVE A DISPOSICIÓN

desde un punto de vista de seguridad. El punto exacto en el proceso de carga dónde la línea divisoria entre el estado 1 y el estado 2 es arrastrada es muy específico a las circunstancias (por ejemplo, ¿A qué código del cargador se parece? ¿Qué tarea necesita ser hecha como parte de la carga?), y representa una clase de intercambio: Por un lado, la línea divisoria debe estar bastante tarde, para que las acciones futuras, la cantidad de variación en la conducta de carga legítima sea bastante pequeña que pueda distinguirse el comportamiento legítimo factiblemente de la conducta ilegítima. (Como se notó anteriormente, en el procedimiento de carga temprano, el número grande de opción que las ROMs y otras variables hacen el número de posibles caminos de ejecución tan grande que es difícil distinguir las conductas legítimas de las conductas ilegítimas.). Por otro lado, la línea divisoria debe estar bastante temprana que preceda cualquier evento (por ejemplo, la carga del programa de autenticación) eso podría tener un efecto en la seguridad, como causar que las claves sean distribuidas inapropiadamente. En general, la línea divisoria entre el estado 1 y el estado 2 permite primero ejecutar un sistema arbitrario en un "raro" (o "abierto", o "inválido") estado durante algún tiempo, y entonces el inicio de cualquiera componente es necesarios para verificar el comportamiento (por ejemplo, un ambiente de alta-seguridad), y entonces usa tales componentes para validar el estado actual de la máquina (o fuerza la máquina a un estado bueno conocido) a tal punto que el proceso de hacer algo puede afectar la seguridad después (por ejemplo, la distribución de claves) puede permitirse proceder.

### Validación 450

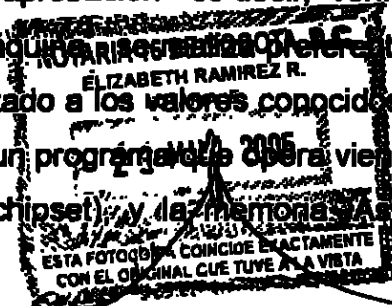
[0043] La validación 450 es, en esencia, el acto de verificar que el sistema operativo cargador 410 (o, más específicamente, una personalización preferida, verificar ese estado 2 del cargador) es un programa confiable conocido, y asegura que la máquina en que está corriendo está en un estado bueno conocido. Así, la aprobación tiene dos partes: (1) examinar el programa del cargador y/o parte 2 del



cargador de programan) para asegurar que es el conocido, el programa confiado que se cree que es, y (2) cambiar el estado de la máquina pertinente a un estado bueno conocido bajo el cual el cargador se conoce que se comporta correctamente. La premisa de realizar (1) y (2) es que un programa conocido, operando en un estado de la máquina conocido, se comportará de una manera conocida.

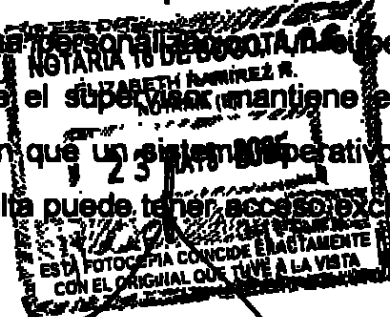
[0044] La primera parte de la validación - es decir, examinar el cargador - puede realizarse en una variedad de maneras. En el ejemplo más simple, puede haber un detalle del programa conocido (qué pueden ser criptográficamente firmados para crear una firma digital, o sino almacenados de alguna manera no-engañada) contra la cual el programa real que está corriendo puede compararse. El problema con tomar un detalle sobre el programa completo es que casos diferentes de un programa legítimo pueden tener imágenes ligeramente diferentes, para que un detalle que se calcula que en cierto modo requiere la identidad completa entre el programa conocido y el caso corriente del programa puede ser demasiado restrictivo. Preferiblemente, la aprobación se ha realizado en cierto modo que asegura que el programa es lo que se supone que es, pero no es demasiado restrictivo - por ejemplo, el proceso de aprobación puede comparar detalles de componentes del programa que se conocen para ser arreglados, y puede realizar otras pruebas en componentes del programa que pueden cambiar con el tiempo. El componente que realiza la aprobación debe detallarse para las especificidades del programa que está siendo validada; la invención no se limita a cualquier técnica particular para validar un programa.

[0045] La segunda parte de la aprobación - es decir, verificar/configuración (verifying/setting) del estado de la máquina, se realiza preferentemente poniendo todas las "fuentes" pertinentes de estado a los valores conocidos. Típicamente, el estado pertinente que puede afectar un programa que opera viene de tres fuentes: la CPU, el chip de configuración (chipset), y la memoria. Así, estos artículos



pueden ponerse en un estado bueno conocido en el momento de aprobación - por ejemplo, la CPU puede ponerse en un estado conocido - por ejemplo, anillo 0, con el contador del programa apuntado a una situación conocida, que todo los datos se registran a cero, y toda la memoria de otra manera que la memoria en que el programa del cargador se almacena para poner a cero. Si el cargador se ha verificado para comportarse correctamente cuando corre con la máquina en este estado, entonces la combinación de examinar al cargador y poner la máquina en el estado bueno conocido debe asegurar la conducta correcta al tiempo en que el programa de autenticación 418 se ejecuta. Como se discutió anteriormente, uno de los beneficios (aunque no el único beneficio) del comportamiento correcto del funcionamiento del programa de autenticación 418 es que los componentes falsos no se cargarán porque causaría que la clave 204 sea distribuido bajo circunstancias impropias.

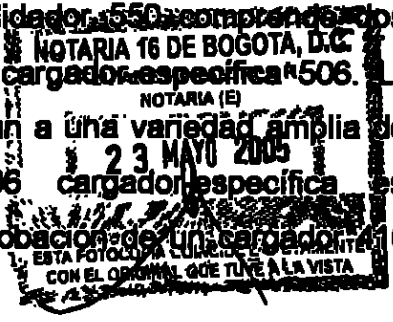
[0046] Debe notarse que, para que el proceso de aprobación proteja la carga del sistema operativo de los componentes falsos, debe haber convicción suficiente para que el proceso propio de aprobación este realizándose correctamente. La exactitud del proceso de aprobación puede asegurarse incluyendo el validador como un agente confiado que corre sobre o en un ambiente de cómputo de alta-seguridad. Puede haber por ejemplo, un pequeño, sistema operativo que realiza un conjunto limitado de funciones, pero que proporciona un alto grado de convicción tal, que un sistema operativo realizará según sus especificaciones. Tal sistema operativo podría ejecutarse junto con otros sistemas operativos en una sola máquina, y la independencia de tal sistema operativo de alta seguridad de otros ambientes (menos seguro) en el sistema podrían ser forzados por un componente de supervisión, tal como un supervisor o monitor de máquina virtual. (En una persona, un supervisor da fuerza a particiones que son ambientes que el supervisor mantiene en algún estado de aislamiento mutuo de nosotros y en que un sistema operativo puede correr.) Es más, el componente de seguridad alta puede tener acceso exclusivo a una raíz de



confianza, como un módulo de hardware que aplica, y fuertemente puede proteger, las claves criptográficas por-plataforma. El validador podría ser un programa (o "agente") que corre en tal ambiente de alta seguridad que proporcionaría la seguridad de la magnitud que el propio ambiente de alta-seguridad puede confiarse para resistir los ataques - el validador no estaría sujeto a los ataques de afuera del ambiente de alta-seguridad que causaría que se comportase incorrectamente. Adicionalmente, como es notado sobre el ambiente de alta-seguridad podría proporcionar el almacenamiento sellado (es decir, la habilidad de guardar algún dato y sólo soltar ese dato a un objeto particular para el cual el dato se ha sellado), y el validador podría usar este almacenamiento sellado para guardar cualquier clave que será distribuida. Debe notarse que el validador puede ser una parte íntegra del ambiente de alta-seguridad, pero también podría ser un componente separado que corre en el ambiente de alta-seguridad.

[0047] La FIG. 5 muestra un ejemplo de un validador 550 que realiza la aprobación de acuerdo con el procedimiento descrito anteriormente. El validador 550 tiene la habilidad de evaluar un cargador 410 para la exactitud y/o complacencia con algún conjunto de normas conocidas. Adicionalmente, el validador 550 tiene la habilidad para evaluar y/o afectar el estado de la máquina 502. Usando la combinación de estas habilidades, el validador 550 puede asegurar que el cargador 410 se comportará correctamente, asegurando que el cargador 410 que se espera que esté corriendo está, y asegurando que la máquina en que está corriendo está en un estado en que el cargador 410 se conoce para comportarse correctamente.

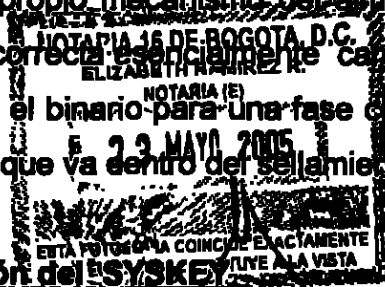
[0048] Estructuralmente, el validador 550 comprende dos componentes: una parte general 504, y una parte cargador-específica 506. La parte general "504" comprende código que es común a una variedad amplia de validadores (o todo el validador). La parte 506 cargador-específica es código que específicamente se relaciona a la aprobación de un cargador 410 particular - es



decir, el código que entiende que es una instancia correcta del cargador 410 (o estado 2 del cargador 410) parecería como si realizara pruebas para asegurar que el cargador 410 conforma esta comprensión. Así, la parte 506 cargador-específica puede combinarse con la parte 504 general para formar un validador completo.

[0049] El validador 550 es el componente que finalmente decide si la clave 204 se proporcionará al cargador (qué lo proporcionará entonces después al sistema operativo, para ser usado por el sistema operativo de acuerdo con la manera permitida por un programa de autenticación, como el descrito anteriormente). El validador 550 puede proteger la clave 204 sellando la clave 204 al validador 550 en una facilidad de almacenamiento sellado 508. La facilidad de almacenamiento sellado 508 puede ser un rasgo de alta-seguridad que calcula el ambiente. La facilidad de almacenamiento sellado 508 puede permitir a los componentes que corren en un ambiente de alta-seguridad para sellar los datos arbitrarios, tal que ningún otro componente puede recuperar los datos. Por ejemplo, la parte 506 cargador-específica del validador 550 puede sellar la clave 204 a sí mismo. La facilidad de almacenamiento sellado 508 impedirá a cualquier componente de otra manera que la parte 506 cargador-específica abra la clave 204. Así, la clave 204 es protegida para que sólo pueda obtenerse de la parte 506 cargador-específica, y la parte 506 cargador-específica distribuirá sólo la clave 204 a un cargador de la parte 506 cargador-específica que no causará que la clave 204 sea distribuida impropriamente. En otra personalización, la clave se sella basada en una combinación de componentes que incluyen el validador y la parte 2 de cargador 410. En tal personalización, la parte cargador-específica del validador 506 puede eliminarse, porque el propio mecanismo del almacenamiento sellado asegura la presencia de una fase correcta esencialmente de cargador 2 que antes de que la clave pueda abrirse, desde el binario para una fase correcta conocida 2 el cargador es la parte de la métrica que va dentro del sellamiento.

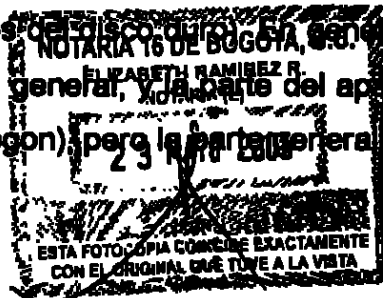
Proceso de ejemplo de Protección del SYSKEY



[0050] La Fig. 6 muestra un proceso del ejemplo de proteger una clave de acuerdo con los aspectos de la invención descritos anteriormente. En algún punto después de que una máquina se inicia, un cargador del sistema operativo (602) se ejecuta. Durante la ejecución del cargador del sistema operativo, se validan el cargador y el estado de la máquina (604), asegurando por eso que el cargador se comportará de una manera predecible por las razones descritas anteriormente. Después de que el cargador y el estado de la máquina se han validado, el cargador se usa para cargar el sistema operativo mientras impide a los componentes falsos ser cargados (606). (Como discutido anteriormente, la validación del cargador y del estado de la máquina significa que el comportamiento futuro del cargador es conocida no para producir la carga de componentes falsos.) Después de que el sistema operativo está cargado, el programa de autenticación se ejecuta, el cual conduce el acceso a la clave.

#### Otras Personalizaciones de Ejemplo

[0051] Debe notarse que no pueden usarse los mecanismos de la invención presente meramente para realizar una carga del sistema operativo, pero generalmente puede usarse para permitir que un sistema realice alguna función en un estado raro (inválido), mientras requiere de la aprobación para proceder con ciertas funciones. Por ejemplo, una computadora puede realizar algunas funciones que pueden realizarse sin ninguna clase de aprobación (por ejemplo, la computadora puede actuar como una radio), pero la aprobación puede tener que ser realizada antes de que la computadora pueda realizar las funciones más sensibles (por ejemplo, leer los archivos del disco duro). En general, una máquina puede verse como un aparato y un PC general, y la parte del aparato no requiere la validación/autenticación (validation/login), pero la parte general del PC la hace.

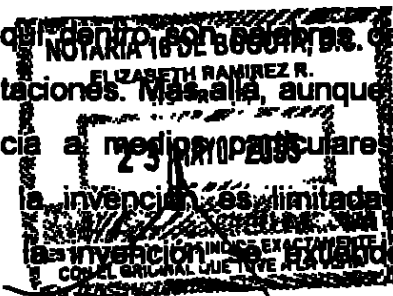


**[0052]** Es más, el validador no se limita a cualquier liberación/no liberación (releasing/not) del conjunto de claves del programa de autenticación, pero generalmente podría ser configurado para liberar conjuntos específicos de claves para pilas de software particulares - por ejemplo, hay una pila de software que puede conseguir una primera clave, pero el validador sólo podría liberar una segunda clave a una pila del "software difícilmente validada."

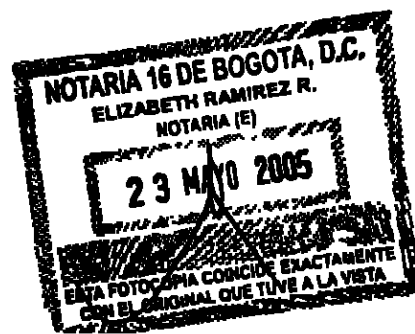
**[0053]** Adicionalmente, las claves necesariamente no necesitan ser liberadas a través de un procedimiento de autenticación, sino que puede proporcionarse a través de los procedimientos de aprobación arbitrarios. Por ejemplo, un jugador de DVD binario podría conseguir, por decir, las claves de función del DVD después de que el validador ha determinado que el jugador es el binario correcto, pero sin requerir una autenticación.

**[0054]** Adicionalmente, algunos programas podrían trabajar en alguna forma sin la aprobación, pero entonces requiere un tipo de aprobación antes de que le permita exponer cierta funcionalidad. Por ejemplo, la aplicación de la telefonía podría empezar sin autenticación requerida, pero puede requerir una autenticación antes de que las claves puedan distribuirse que permitiría a las funciones criptográficas de la aplicación trabajar.

**[0055]** Es notado que los ejemplos anteriores se han proporcionado meramente con el propósito de explicación y no han estado de ninguna manera para ser traducidos como limitantes de la invención presente. Mientras la invención se ha descrito con referencia a varias personalizaciones, se entiende que las palabras que se han usado aquí dentro son palabras de descripción e ilustración, en lugar de palabras de limitaciones. Mas allá, aunque la invención se ha descrito aquí dentro con referencia a medios particulares, materiales y personalizaciones, no se piensa que la invención es limitada a los detalles descubiertos aquí dentro; más bien, la invención se extiende a todas las



estructuras funcionalmente equivalentes, métodos y usos, como está dentro del alcance de las demandas adjuntas. Esos experimentados en el arte, teniendo el beneficio de enseñar esta especificación, pueden efectuar numerosas modificaciones y cambios de esto sin partir del alcance y espíritu de la invención en sus aspectos.



**Lo que se Reivindica:**

1. Un medio legible por computadora codificado con las instrucciones legibles por computadora para realizar un método comprendiendo:

iniciar un cargador del sistema operativo;

validar la identidad o exactitud de dicho cargador;

asegurar que una máquina en la que dicho cargador de sistema operativo corre en un estado conocido; y

si la identidad o exactitud de dicho cargador se valida, y si dicha máquina en la cual corre el sistema operativo está en un estado conocido, entonces,:

proporcionar una clave a dicho cargador; y

permitir que dicho cargador cargue un sistema operativo.

2. El medio legible por computadora de la demanda 1, en donde el método más allá comprende:

ejecutar un programa de autenticación que conduzca el acceso a la clave.

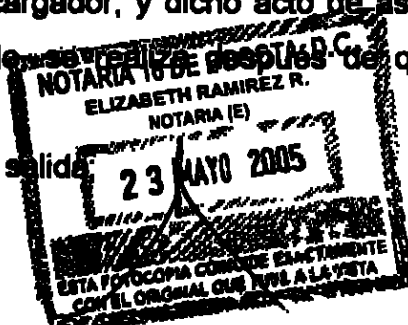
3. El medio legible por computadora de la demanda 2, en que dicho programa de autenticación asegura que un usuario proporcione las credenciales como una condición a dicho programa de autenticación concediendo el acceso a la clave.

4. El medio legible por computadora de la demanda 1, en que dicha clave es útil para descifrar una partición del sistema operativo cifrada.

5. El medio legible por computadora de la demanda 1, en que dicho acto de validar la identidad o exactitud de dicho cargador, y dicho acto de asegurar que una máquina está en un estado conocido, se realiza después de que todo lo siguiente se ha ejecutado:

un sistema básico de entrada salida;

una opción ROM;



un registro de arranque principal; y  
un sector de arranque.

6. El medio legible por computadora de la demanda 5, en qué dicho acto de validar la identidad o exactitud de dicho cargador, y dicho acto de asegurar que una máquina está en un estado conocido, se realiza después de que una parte del cargador del sistema operativo dicho se ha ejecutado.

7. El medio legible por computadora de la demanda 6, en qué dicho acto de validar se realiza después de que todo o substancialmente todo un sistema operativo ha empezado, o después de que algún número de particiones ha empezado, cada una de las particiones que comprenden un ambiente de algún grado de aislamiento de otras particiones es mantenida por un supervisor.

8. El medio legible por computadora de la demanda 1, en qué dicha clave es sellada para un validador que realiza dicha validación asegurando el acto y dicho acto de aseguramiento, en donde el método más allá comprende:  
dicho validador abriendo dicha clave.

9. El medio legible por computadora de la demanda 8, en qué dicha clave es sellada por lo menos a dicho validador, y en qué dicho validador valida una parte de por lo menos dicho cargador.

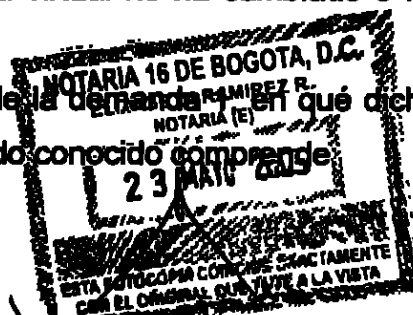
10. El medio legible por computadora de la demanda 8, en qué dicha clave es sellada al validador dicho y a por lo menos una parte de dicho cargador.

11. El medio legible por computadora de la demanda 1, en qué dicha máquina comprende una máquina física.

12. El medio legible por computadora de la demanda 1, en qué dicha máquina comprende una máquina virtual.

13. El medio legible por computadora de la demanda 12, en qué dicha clave sólo se revela si una arquitectura de máquina virtual no ha cambiado o ha sido válida.

14. El medio legible por computadora de la demanda 1, en qué dicho acto de asegurar que una máquina está en un estado conocido comprende



evaluar un estado actual de máquina y comparar dicho estado actual con el estado conocido dicho.

15. El medio legible por computadora de la demanda 1, en qué dicho acto de asegurar que una máquina está en un estado conocido comprende:

configurar un estado actual de máquina para ser consistente con dicho estado conocido.

16. El medio legible por computadora de la demanda 1, en qué dicho cargador es un sistema operativo completo, o una instancia completa de un sistema operativo.

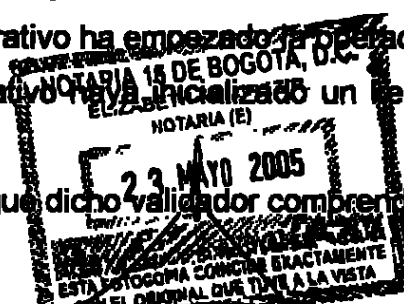
17. Un sistema para realizar una arranque de un sistema operativo bajo circunstancias que proporcionan las convicciones acerca de la fiabilidad del arranque, el sistema comprende:

un validador que evalúa la exactitud o identidad de un cargador del sistema operativo que cargará el sistema operativo, y ese además evalúa un estado de una máquina en que dicho cargador del sistema operativo operará, el cual permite o no al cargador del sistema operativo para proceder con cargar el sistema operativo que depende de si o no la exactitud o identidad del cargador del sistema operativo se verifica, y qué lugares de dicha máquina estuvieron en un estado anterior conocido para permitir que dicho cargador del sistema operativo proceda.

18. El sistema de la demanda 17, en donde una clave es sellada a dicho validador, y en donde dicho validador abre dicha clave y proporciona dicho clave al cargador del sistema operativo dicho si dicho cargador del sistema operativo es permitido para proceder.

19. El sistema de la demanda 17, en qué dicho validador evalúa la exactitud o identidad del cargador del sistema operativo, y evalúa el estado de la máquina, después de que el cargador del sistema operativo ha empezado la operación, pero antes de que el cargador del sistema operativo haya inicializado un kernel o un controlador del dispositivo.

20. El sistema de la demanda 17, en que dicho validador comprende:



una parte general que es común a una clase de validador; y

una parte específica que es específico al cargador del sistema operativo dicho, y eso es reemplazable con una parte diferente en el caso dónde dicho validador está usándose para validar a un cargador del sistema operativo diferente.

21. El sistema de la demanda 17, en qué dicha máquina comprende una máquina física.

22. El sistema de la demanda 17, en qué dicha máquina comprende una máquina virtual.

23. Un método de arranque de un sistema operativo comprende:

ejecutar una entrada salida básica del sistema, una opción ROM, un registro de arranque principal, y un sector de arranque;

empezar a un cargador del sistema operativo;

validan dicho cargador del sistema operativo;

validar un estado de una máquina en que dicho cargador del sistema operativo se ejecuta;

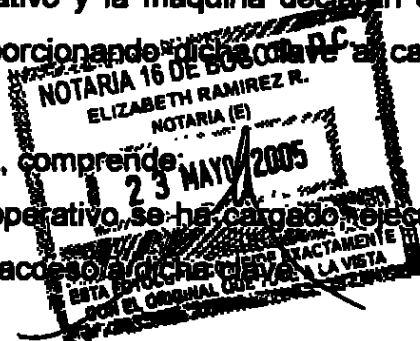
si dicho cargador del sistema operativo, y dicho estado de máquina, se determina para ser válido, permitiendo entonces que dicho cargador del sistema operativo para cargue un sistema operativo.

24. El método de la demanda 23, en donde una clave que es necesaria realizar correctamente por lo menos una función bajo el sistema operativo dicho se sella a un validador que realiza dichos actos de validar a un cargador del sistema operativo y validar un estado de la máquina, y en donde el método más allá comprende:

si el cargador del sistema operativo y la máquina declaran que es válido, abriendo entonces dicha clave y proporcionando dicha clave al cargador del sistema operativo dicho.

25. El método de demanda 23, más allá, comprende:

después de que dicho sistema operativo se ha cargado, ejecutando un programa de autenticación que conduce el acceso a dicha clave.



26. El método de la demanda 23, en qué o dicho programa de autenticación permite o no que componentes de dicho sistema operativo usen dicha clave que depende de si un usuario completa un procedimiento de autenticación con éxito.

27. El método de demanda 23, en qué dichos actos de validar dicho cargador del sistema operativo y validando dicho estado de la máquina se realiza después de que el cargador del sistema operativo dicho ha realizado por lo menos una acción.

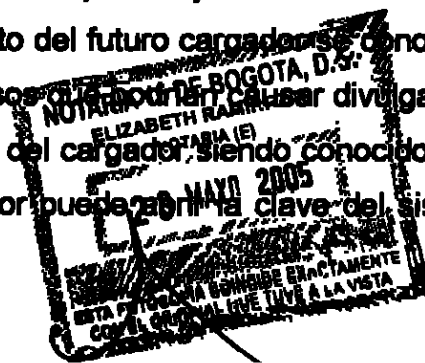
28. El método de la demanda 27, en qué dichos actos de validar dicho cargador del sistema operativo y validando el estado de dicha máquina se realiza a tiempo antes de restablecer el estado de la máquina que le impedirá al cargador funcionar para cargar correctamente dicho el sistema operativo.

29. El método de la demanda 23, en qué dicha máquina comprende una máquina física.

30. El método de la demanda 23, en qué dicha máquina comprende una máquina virtual.

### **RESUMEN DEL DESCUBRIMIENTO**

Un mecanismo para arranque del sistema operativo protegido que impide a los componentes falsos ser cargados con el sistema operativo, y además prevenir la divulgación de la clave del sistema bajo circunstancias impropias. Después de que una parte del procedimiento de inicio (startup) de máquina ha ocurrido, el cargador del sistema operativo se ejecuta, el cargador se valida, y un estado de la máquina correcto o se verifica para existir y/o se crea. Una vez el cargador se ha verificado que es un cargador legítimo, y el estado de la máquina bajo el cual está corriendo se verifica que es correcto, el comportamiento del futuro cargador se conoce para proteger contra la carga de componentes falsos que podrían causar divulgación de la clave del sistema. Con el comportamiento del cargador, siendo conocido que es seguro para la clave del sistema, el validador puede emitir la clave del sistema y puede proporcionarlo al cargador.



Con encabezado de envío de fax Jun 30 2004 10:30 FR Microsoft Building 40 425  
936 7329 Para 912155683439 P07/09

## SOLICITUD DE PATENTE

### DECLARACIÓN Y PODER

Expediente del Abogado No. MSFT-3013      Expediente de MS No. 305832.01

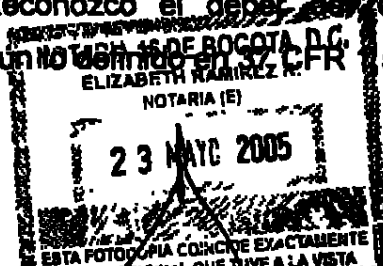
Como el inventor nombrado a continuación, por medio del presente declaro que:  
Mi residencia/dirección postal y ciudadanía son los indicados a continuación  
debajo de mi nombre;

Creo ser el inventor original, primero y único (en caso de haber sólo un nombre  
indicado abajo), o el inventor original, primero y en conjunto (en caso de haber  
varios nombres indicados abajo) del objeto reivindicado y por el cual se solicita  
una patente sobre el invento llamado SISTEMA Y MÉTODO PARA INICIO DE  
SISTEMA OPERATIVO PROTEGIDO UTILIZANDO VALIDACIÓN DE LA  
CONDICIÓN

la especificación del cual se presenta anexa al presenta, salvo que la siguiente  
casilla se encuentra marcada:

( ) fue presentado el \_\_\_\_ como Número de Serie de Solicitud de Estados  
Unidos o Número de Solicitud Internacional PCT \_\_\_\_ modificado el  
\_\_\_\_ (si aplica).

Declaro que he revisado y entendido los contenidos de la especificación  
previamente mencionada, incluyendo las reivindicaciones, en lo modificado por la  
(s) modificación (es) referidas arriba. Reconozco el deber de revelar toda  
información esencial a la patentabilidad según lo definido en 37 CFR 1.56.



Solicitud (es) Extranjeras y/ Reivindicación de Prioridad en el Extranjero

Por el presente reivindico los beneficios de prioridad en el extranjero según el Título 35 de la sección 119 del Código de los Estados Unidos, sobre cualquier solicitud de patente o certificado de inventor indicado a continuación y he identificado a continuación cualquier solicitud de patente o inventor (es) certificados con una fecha de presentación previa a la solicitud sobre el cual se reivindica la prioridad:

| PAIS | NO. SOLICITUD | FECHA DE PRESENTACIÓN | PRIORIDAD REIVINDICADA SEGÚN 35 U.S.C. 119 |
|------|---------------|-----------------------|--------------------------------------------|
|      |               |                       | SI: ____ NO:                               |
|      |               |                       | SI: ____ NO:                               |

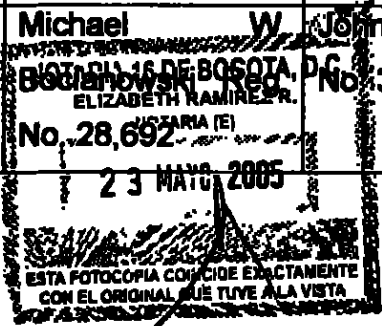
PODER:

Como inventor, designo por medio del presente el (los) siguiente (s) abogado (s) y/o agente (s) asociados con

Código de Barras **23377**

y

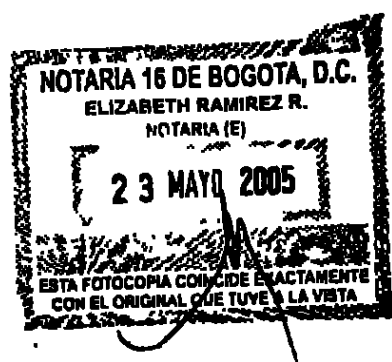
|                                         |                                   |                                    |                                   |
|-----------------------------------------|-----------------------------------|------------------------------------|-----------------------------------|
| David Bartley Eppenauer Reg. No. 35,499 | Martin L. Shively Reg. No. 33,553 | Ronald O. Zink Reg. No. 35,744     | Patricia E. Bomes Reg. No. 37,038 |
| Stacy Quan Reg. No. 33,760              | Jeffrey L. Ranck Reg. No. 38,590  | Michael W. Douglas Reg. No. 28,692 | John Weresh Reg. No. 32,332       |



para hacer seguimiento a esta solicitud y llevar a cabo todos las transacciones con la Oficina de Patentes y Marcas Registradas relacionadas con el mismo.

|                                                        |                                    |
|--------------------------------------------------------|------------------------------------|
| Enviar correspondencia a:                              | Dirigir llamadas a:                |
| Contacto: Steven J. Rocci                              | Contacto: Peter M. Ullman          |
| Nombre de la Firma: Woodcock Washburn LLP              | Teléfono de Contacto: 215-568-3100 |
| Dirección de la Firma: One Liberty Plaza, 46th Floor   |                                    |
| Ciudad, Estado y Código Postal: Philadelphia, PA 19103 |                                    |

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005



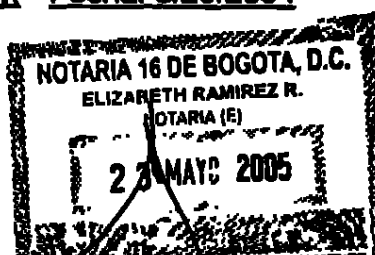
**OCTAVA PÁGINA:** Con encabezado de envío de fax Jun 30 2004 10:30 FR  
Microsoft Building 40 425 936 7329 Para 912155683439 P08/09

**SOLICITUD DE PATENTE  
DECLARACIÓN Y PODER**

Expediente del Abogado No. MSFT-3013      Expediente de MS No. 305832.01  
Declaro que todas las afirmaciones aquí establecidas son en mi conocimiento verdaderas y que todas las afirmaciones están basadas en información y creencias que se cree son verdaderas; adicionalmente, todas las afirmaciones fueron elaboradas con el conocimiento de que declaraciones falsas intencionales y relacionadas son castigados con multa o prisión o ambos, según Sección 1001 del Título 18 del Código de los Estados Unidos, y que dichas declaraciones falsas intencionales pueden poner en peligro la validez de la solicitud o cualquier patente otorgado por ésta.

Nombre del Inventor: Bryan Mark Willman      Ciudadanía: Estados Unidos  
Residencia: Kirkland, Washington      de América  
Dirección de la Oficina: 10901 – 80th Place      Fecha: (en blanco)  
NE. Kirkland, Washington 98034  
(Firma en blanco)  
Firma del Inventor

Nombre del Inventor: Paul England      Ciudadanía: Reino Unido  
Residencia: Bellevue, Washington  
Dirección de la Oficina: 16659 Northup Way      Fecha: 6/29/2004  
Bellevue, Washington 98008  
(Firma)  
Firma del Inventor



Nombre del Inventor: Kenneth D. Ray

Ciudadanía: Estados Unidos

Residencia: Seattle, Washington

de América

Dirección de la Oficina: 928 North 34th Street # 88, Fecha: 6/30/2004

Seattle, Washington 98103

(Firma)

Firma del Inventor

Nombre del Inventor: Jamie Hunter

Ciudadanía: Reino Unido

Residencia: Bothell, Washington

Dirección de la Oficina: 14226 93rd Avenue NE,

Fecha: 6/29/2004

Bothell, Washington 98011

(Firma)

Firma del Inventor

Nombre del Inventor: Lonny Dean McMichael

Ciudadanía: Estados Unidos

Residencia: Redmond, Washington

de América

Dirección de la Oficina: 9950 229th Ln NE,

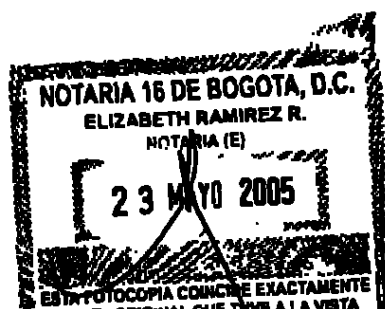
Fecha: 6/29/2004

Redmond, Washington 98053

(Firma)

Firma del Inventor

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 03/14/2005



**NOVENA PÁGINA:** Con encabezado de envío de fax Jun 30 2004 10:30 FR  
 Microsoft Building 40 425 936 7329 Para 912155683439 P09/09

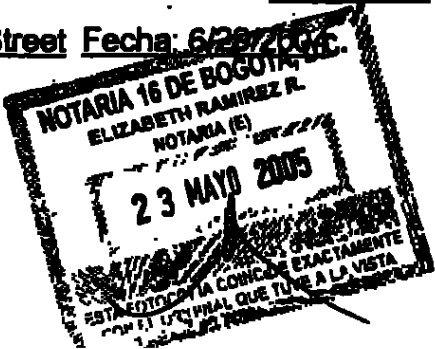
**SOLICITUD DE PATENTE  
 DECLARACIÓN Y PODER**

Expediente del Abogado No. MSFT-3013      Expediente de MS No. 305832.01

Nombre del Inventor: Derek Norman LaSalle      Ciudadanía: Estados Unidos  
 Residencia: Redmond, Washington      de América  
 Dirección de la Oficina: 5509 162nd Avenue NE,      Fecha: (en blanco)  
Redmond, Washington 98052  
 (Firma en blanco)  
 Firma del Inventor

Nombre del Inventor: Pierre Jacomet      Ciudadanía: Francia  
 Residencia: Sammamish, Washington  
 Dirección de la Oficina: 1221 240th Way SE,      Fecha: (en blanco)  
Sammamish, Washington 98075  
 (Firma en blanco)  
 Firma del Inventor

Nombre del Inventor: Mark Eliot Paley      Ciudadanía: Estados Unidos  
 Residencia: Redmond, Washington      de América  
 Dirección de la Oficina: 18112 NE 31st Street      Fecha: 6/28/2004  
Redmond, Washington 98052  
 (Firma)  
 Firma del Inventor



Nombre del Inventor: Thekkthalackal Varugis Kurien Ciudadanía: Estados Unidos  
Residencia: Sammamish, Washington de América  
Dirección de la Oficina: 22120 NE 23rd Street Fecha: 6/29/2004  
Sammamish, Washington 98074

(Firma)

Firma del Inventor

Nombre del Inventor: David B. Cross Ciudadanía: Estados Unidos  
Residencia: Redmond, Washington de América  
Dirección de la Oficina: 27524 NE 31st Court Fecha: 6/29/2004  
Redmond, Washington 98053

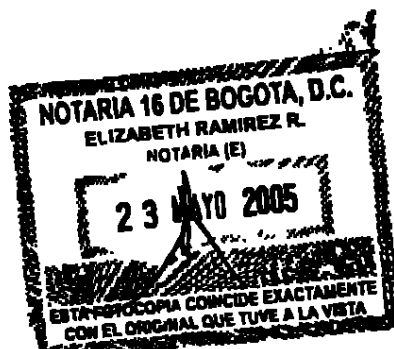
(Firma)

Firma del Inventor

Página 3 de 3

Total de páginas. 09

Nota al final: Copia suministrada por USPTO de la IFW Image Database el  
03/14/2005



162

Ambiente de Cómputo 100

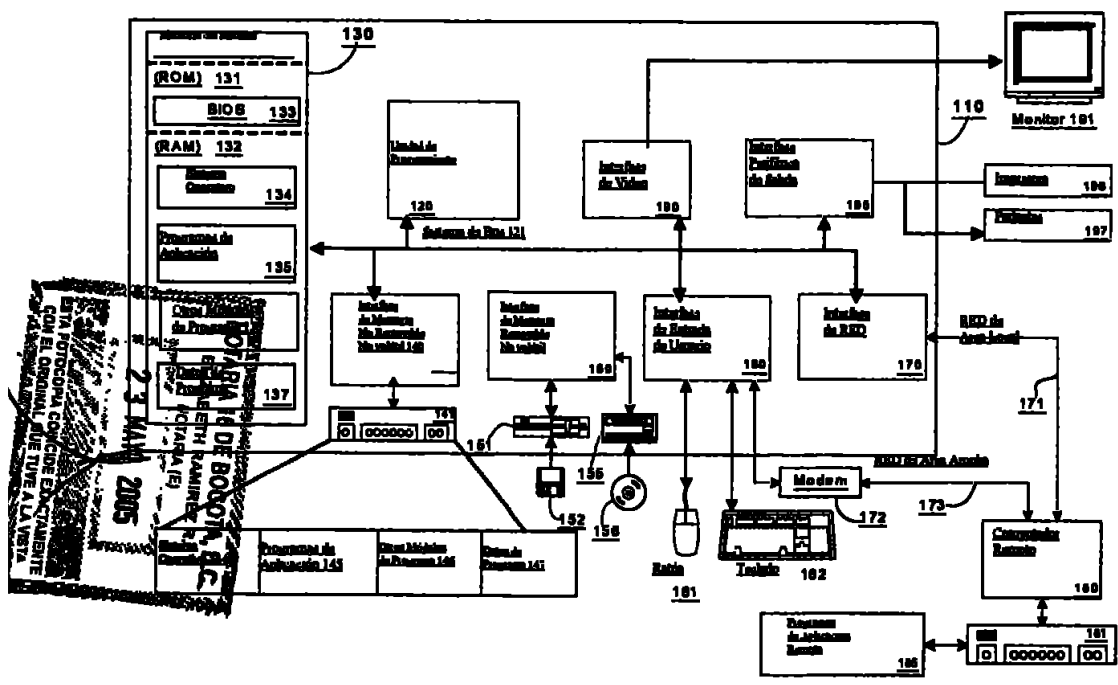


FIG. 1

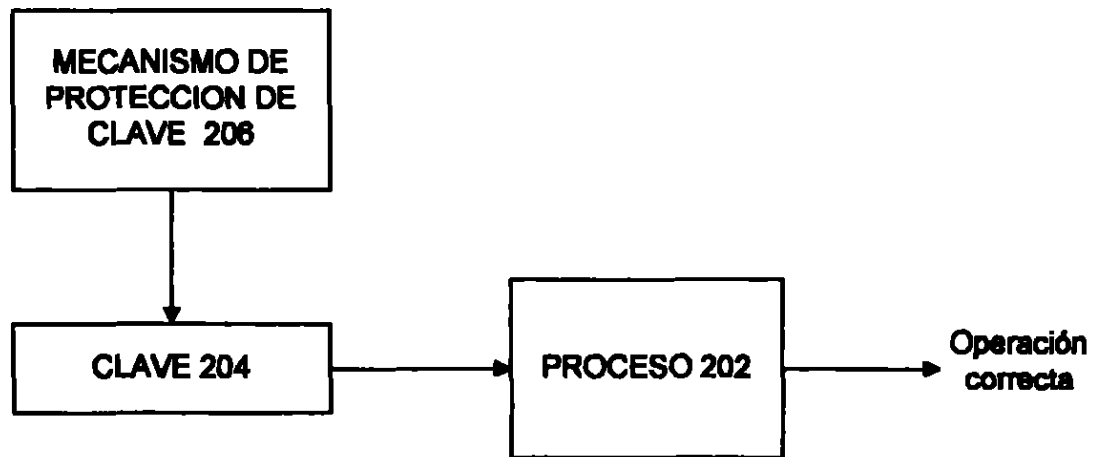
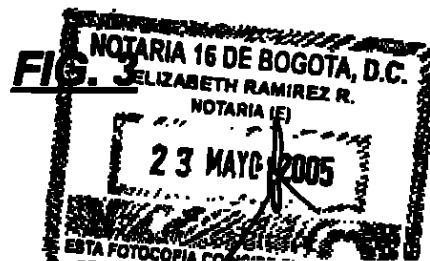
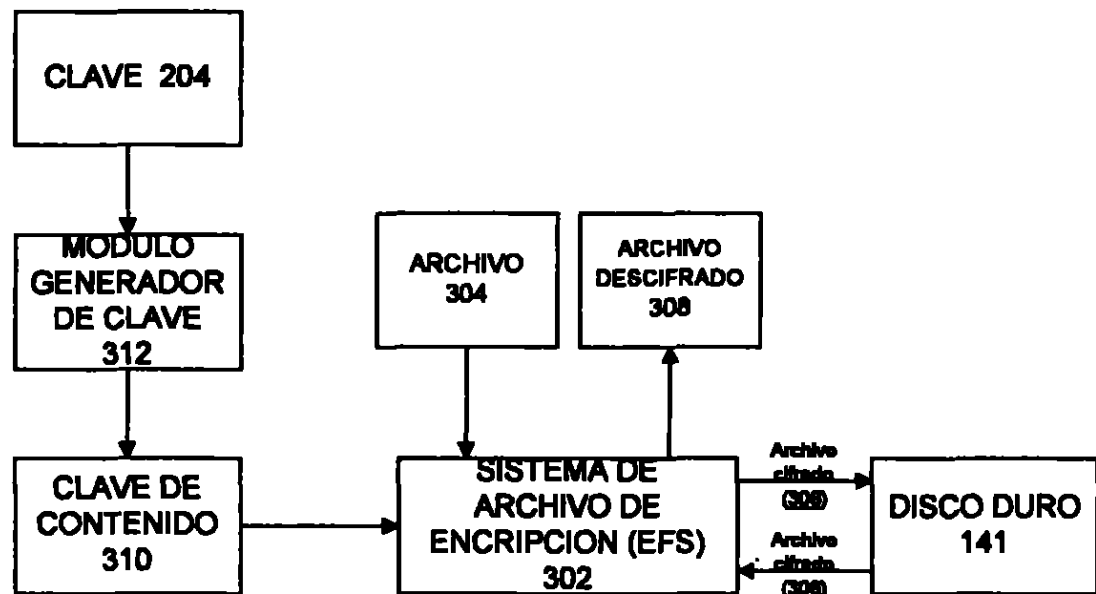


**FIG. 2**



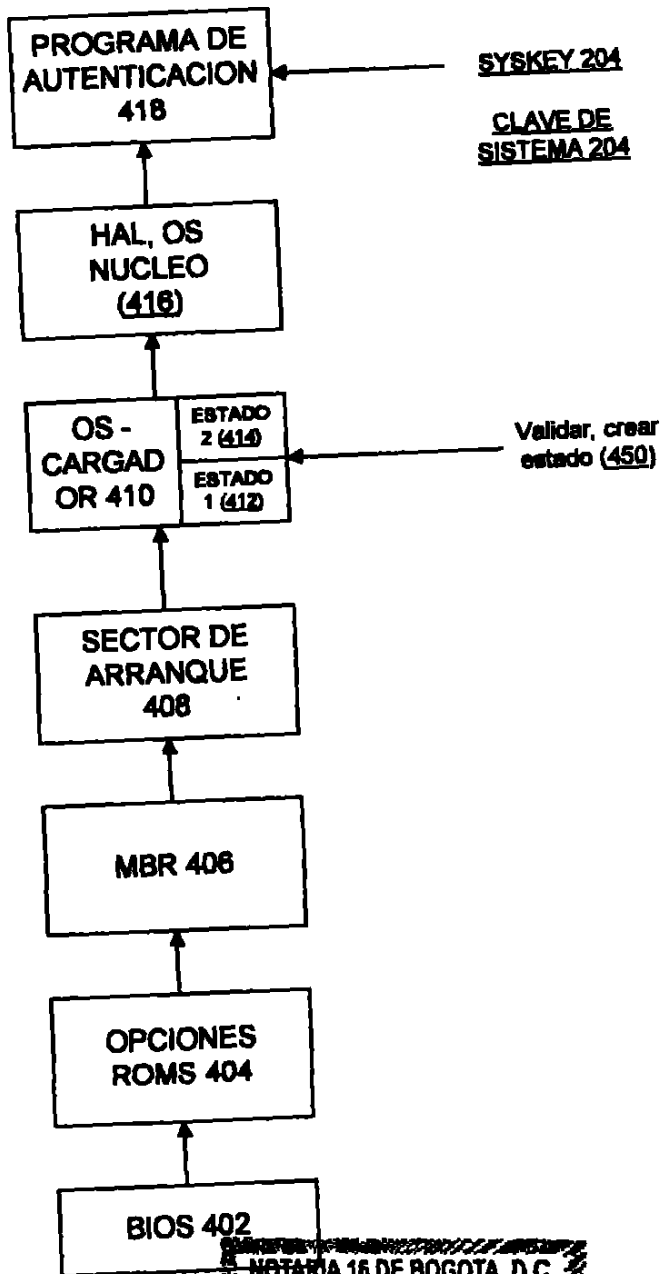
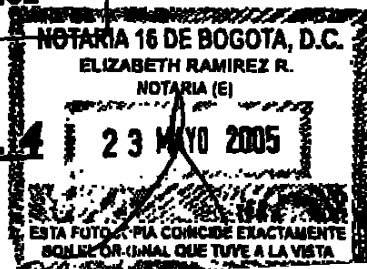


FIG. 4



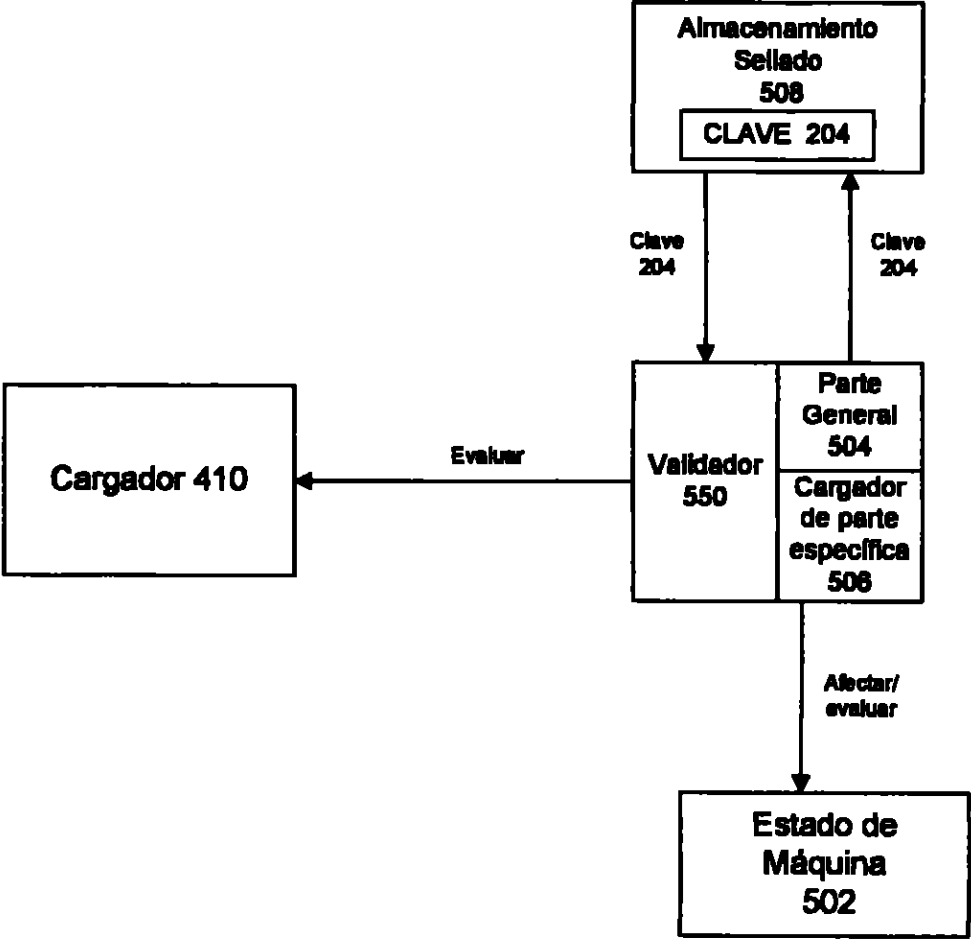
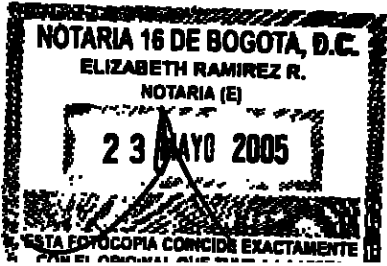


**FIG. 5**



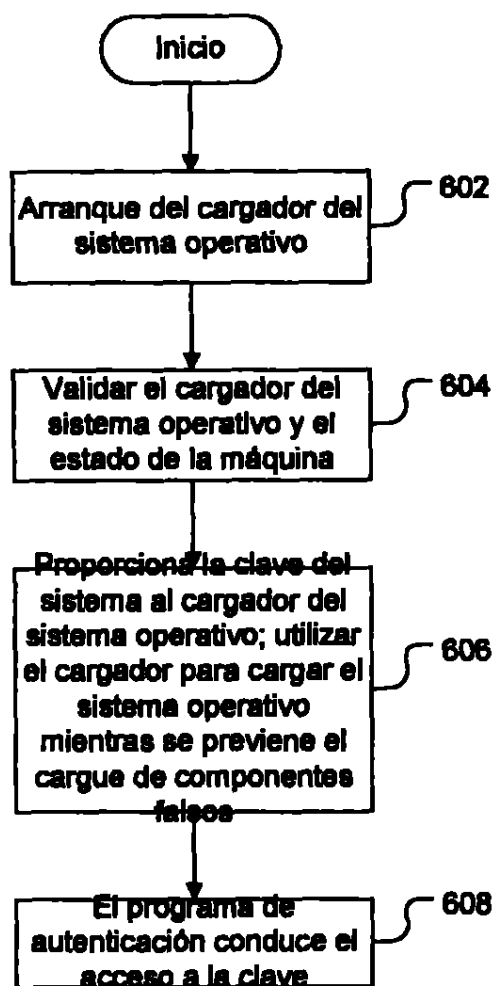
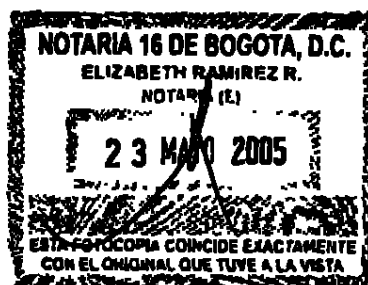


**FIG. 6**



**APOSTILLE**

*(Convention de La Haye du 5 octobre 1961)*

1. Country: United States of America

This public document

2. has been signed by M. K. Hawkins

3. acting in the capacity of Certifying Officer

4. bears the seal/stamp of Patent and Trademark Office

**Certified**

5. at Washington, D.C.

6. the twenty-second of March, 2005

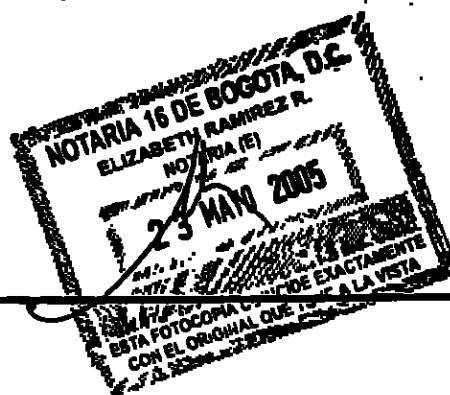
7. by Assistant Authentication Officer, United States Department of State

8. No. 05014224-4

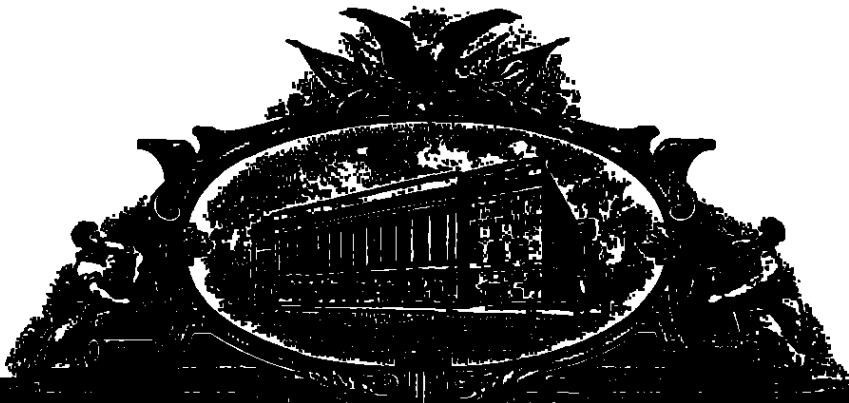
9. Seal/Stamp:

10. Signature:

  
Sonya N. Johnson



A 1296009

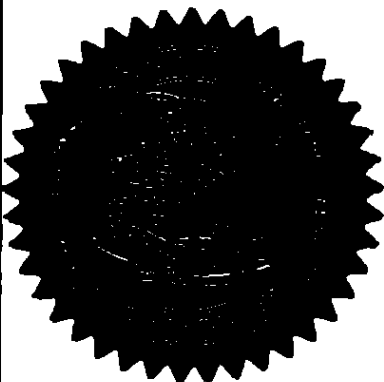


# THE UNITED STATES OF AMERICA

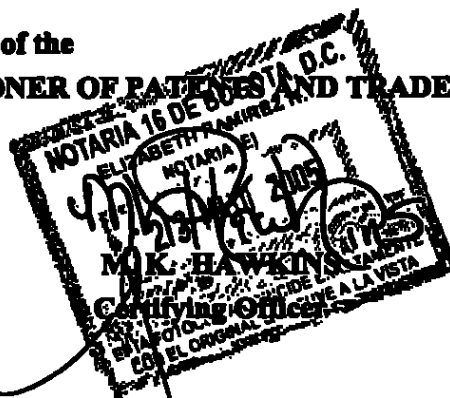
**TO ALL TO WHOM THESE PRESENTS SHALL COME:**  
**UNITED STATES DEPARTMENT OF COMMERCE**  
**United States Patent and Trademark Office**

March 18, 2005

**THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE  
RECORDS OF THIS OFFICE OF A DOCUMENT RECORDED ON  
NOVEMBER 12, 2004.**



By Authority of the  
**COMMISSIONER OF PATENTS AND TRADEMARKS**

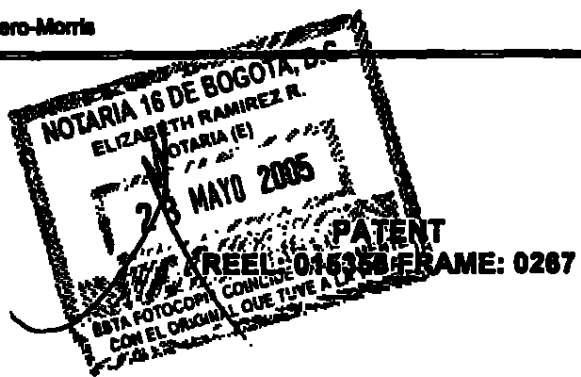


PATENT ASSIGNMENT

Electronic Version v1.1  
 Stylesheet Version v1.1

|                                                                                                                                           |                                  |                         |  |
|-------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|-------------------------|--|
| SUBMISSION TYPE:                                                                                                                          |                                  | NEW ASSIGNMENT          |  |
| NATURE OF CONVEYANCE:                                                                                                                     |                                  | ASSIGNMENT              |  |
| CONVEYING PARTY DATA                                                                                                                      |                                  |                         |  |
| Name                                                                                                                                      |                                  | Execution Date          |  |
| Bryan Mark Wilman                                                                                                                         |                                  | 11/02/2004              |  |
| Pierre Jacomet                                                                                                                            |                                  | 11/02/2004              |  |
| Derek Norman LaSalle                                                                                                                      |                                  | 11/04/2004              |  |
| RECEIVING PARTY DATA                                                                                                                      |                                  |                         |  |
| Name:                                                                                                                                     | Microsoft Corporation            |                         |  |
| Street Address:                                                                                                                           | One Microsoft Way                |                         |  |
| City:                                                                                                                                     | Redmond                          |                         |  |
| State/Country:                                                                                                                            | WASHINGTON                       |                         |  |
| Postal Code:                                                                                                                              | 98062                            |                         |  |
| PROPERTY NUMBERS Total: 1                                                                                                                 |                                  |                         |  |
| Property Type                                                                                                                             | Number                           |                         |  |
| Application Number:                                                                                                                       | 10882134                         |                         |  |
| CORRESPONDENCE DATA                                                                                                                       |                                  |                         |  |
| Fax Number:                                                                                                                               | (215)868-3439                    |                         |  |
| <i>Correspondence will be sent via US Mail when the fax attempt is unsuccessful.</i>                                                      |                                  |                         |  |
| Phone:                                                                                                                                    | 2158683100                       |                         |  |
| Email:                                                                                                                                    | pfilero@woodcock.com             |                         |  |
| Correspondent Name:                                                                                                                       | Woodcock Washburn LLP            |                         |  |
| Address Line 1:                                                                                                                           | One Liberty Place, 48th floor    |                         |  |
| Address Line 4:                                                                                                                           | Philadelphia, PENNSYLVANIA 19103 |                         |  |
| NAME OF SUBMITTER:                                                                                                                        |                                  | Charlotte Filero-Morris |  |
| Total Attachments: 10<br>source=10882134#page1.tif<br>source=10882134#page2.tif<br>source=10882134#page3.tif<br>source=10882134#page4.tif |                                  |                         |  |

500011858



CH \$40.00 10882134

source=10882134#page5.tif  
source=10882134#page6.tif  
source=10882134#page7.tif  
source=10882134#page8.tif  
source=10882134#page9.tif  
source=10882134#page10.tif



PATENT  
REEL: 018366 FRAME: 0268

171

## IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number \_\_\_\_\_ Not Yet Assigned  
 Filing Date \_\_\_\_\_ Herewith  
 Invention \_\_\_\_\_ Bryan Mark Wilman, et al.  
 Applicant \_\_\_\_\_ Microsoft Corporation  
 Attorney's Docket No. \_\_\_\_\_ MSFT-3013/305832.01  
 Title: **SYSTEM AND METHOD FOR PROTECTED OPERATING SYSTEM BOOT  
USING STATE VALIDATION**

## PATENT ASSIGNMENT

## PARTIES TO THE ASSIGNMENT

## Assignors:

Bryan Mark Wilman  
 10701 - 80th Place, NE  
 Kirkland, Washington 98034

Paul England  
 16639 Nordrup Way  
 Bellevue, Washington 98008

Kenneth D. Roy  
 920 North 34th Street, #28  
 Seattle, Washington 98103

Jamie Hunter  
 14225 95th Ave NE  
 Redmond, WA 98011

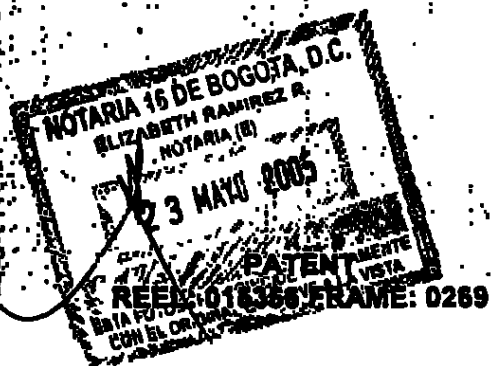
Larry Dean McMichael  
 9030 225th Ln NE  
 Redmond, WA 98053

Derek Norman LaSalle  
 3509 162nd Ave NE  
 Redmond, WA 98052

Pierre Jaccout  
 1221 240th Way, SE  
 Sammamish, Washington 98075

Mark Eric Foley  
 18112 NE 51st Street  
 Redmond, WA 98052

Thekkathackal Varughese Kurian  
 22150 NE 23rd Street  
 Sammamish, Washington 98074



172

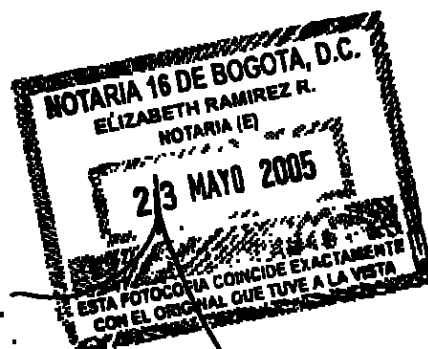
NOV 88 2004 17:42 FR MICROSOFT BUILDING 8

TO 912153683439

P.03/05

David B. Cross  
27324 NE 31st Court  
Redmond, Washington 98053

Page 3 of 5



PATENT  
REEL: 015356 FRAME: 0270

173

**Assignor**

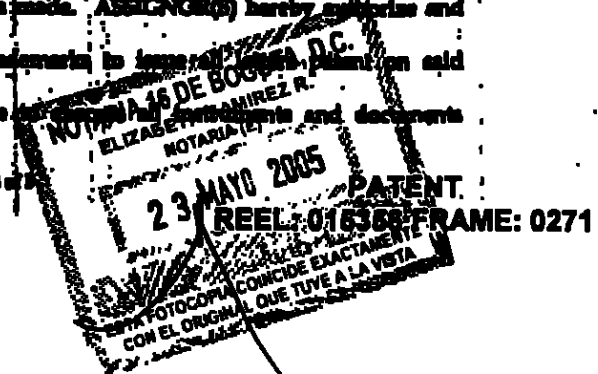
Microsoft Corporation  
 Corporation in the State of Washington  
 One Microsoft Way  
 Redmond, WA 98052

**AGREEMENT**

WHEREAS, ASSIGNOR(S) (listed above) are inventor(s) of an invention entitled as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNOR(S) hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, renewals and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR(S) had this assignment, transfer and sale not been made. ASSIGNOR(S) hereby authorize and request the Commissioner of Patents and Trademarks to issue a patent on said invention to ASSIGNEE. ASSIGNOR(S) agree to execute all forms and documents



required for the making and prosecution of applications for United States and foreign letters patent on said invention, for litigation regarding said letters patent, or for the purpose of protecting title to said invention or letters patent thereafter.



2004-11-02  
Date

Bryan Mark Willman  
Bryan Mark Willman

\_\_\_\_\_  
Date

\_\_\_\_\_  
Paul England

\_\_\_\_\_  
Date

\_\_\_\_\_  
Kenneth D. Ray

\_\_\_\_\_  
Date

\_\_\_\_\_  
Jared Hunter

\_\_\_\_\_  
Date

\_\_\_\_\_  
Larry Dale McMichael

\_\_\_\_\_  
Date

\_\_\_\_\_  
Derek Norman Joffe

2004-11-02  
Date

Philip Jacobson  
Philip Jacobson

\_\_\_\_\_  
Date

\_\_\_\_\_  
Mark Elliot Foley

\_\_\_\_\_  
Date

\_\_\_\_\_  
Theodoros J. Vassiliou-Kirios

\_\_\_\_\_  
Date

\_\_\_\_\_  
David E. Coad

Page 6 of 8  
NOTARIA 16 DE BOGOTA, D.C.  
ELIZABETH RAMIREZ R.  
NOTARIA (E)  
23 MAYO 2005  
REEL: 016388  
FRAME: 0273  
TOTAL PAGE: 06  
CON FOTOCOPIA CONCIDE EXACTAMENTE  
CON EL ORIGINAL QUE TUVE A LA VISTA

# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number..... Not Yet Assigned  
 Filing Date..... Herewith  
 Inventorship..... Bryan Mark Willman, et al  
 Applicant..... Microsoft Corporation  
 Attorney's Docket No. .... MSFT-3013/305832.01  
 Title: SYSTEM AND METHOD FOR PROTECTED OPERATING SYSTEM BOOT  
 USING STATE VALIDATION

## PATENT ASSIGNMENT

### FACTS TO THE ASSIGNMENT

#### Assignors:

Bryan Mark Willman  
 10801 - 80th Place, NE  
 Kirkland, Washington 98034

Paul England  
 16639 Northrup Way  
 Bellevue, Washington 98008

Kenneth D. Ray  
 820 North 34th Street, #88  
 Seattle, Washington 98105

Jamie Hunter  
 14226 93rd Ave NE  
 Bothell, WA 98011

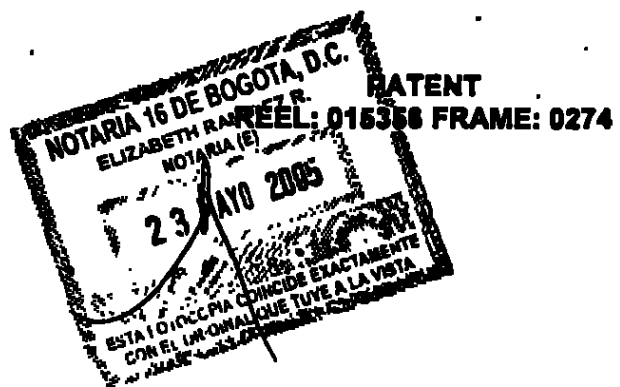
Lanny Dean McMichael  
 9950 229th Ln NE  
 Redmond, WA 98053

Derek Norman LaSalle  
 5509 162nd Ave NE  
 Redmond, WA 98052

Pierre Jacomet  
 1221 240th Way, SE  
 Sumner, Washington 98075

Mark Elliot Paley  
 18112 NE 31st Street  
 Redmond, WA 98052

Thekthalactal Varugis Kurian  
 22120 NE 23rd Street  
 Sumner, Washington 98074



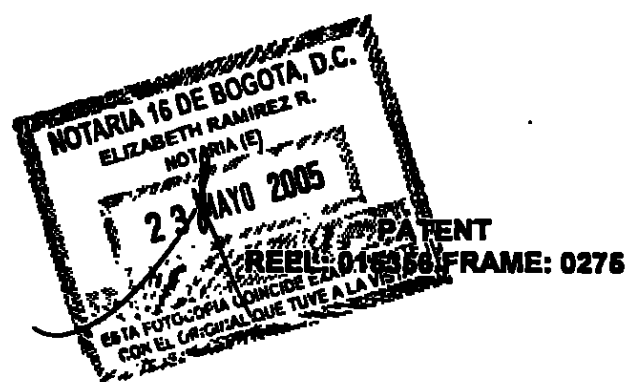
177

NOV 85 2884 17:59 FR MICROSOFT

425 783 2836 TO 912155683439

P.83/86

David B. Cross  
27524 NE 31st Court  
Redmond, Washington 98053



178

**Assignor**

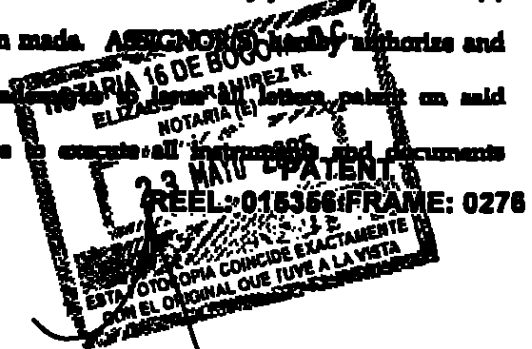
Microsoft Corporation  
Corporation in the State of Washington  
One Microsoft Way  
Redmond, WA 98052

**AGREEMENT**

WHEREAS, ASSIGNOR(S) (listed above) are inventor(s) of an invention entitled as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNOR(S) hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, renewals and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR(S) had this assignment, transfer and sale not been made. ASSIGNOR(S) hereby authorize and request the Commissioner of Patents and Trademarks to issue all letters patent on said invention to ASSIGNEE. ASSIGNOR(S) agree to execute all instruments and documents



179

required for the making and prosecution of applications for United States and foreign letters patent on said invention, for litigation regarding said letters patent, or for the purpose of protecting title to said invention or letters patent therefor.



PATENT  
368 FRAME: 0277

Date

Bryan Mack Williams

Date

Paul England

Date

Kenneth D. Ray

Date

Jamie Hunter

Date

Lenny Dean McMichael

Date

Nov 9th 2004

  
Derek Northing (ASalle)

Date

Pierre Jacomet

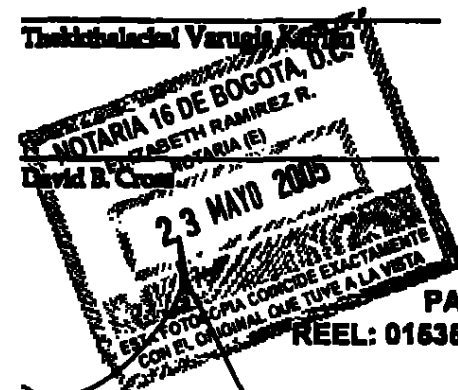
Date

Mark Elliot Paley

Date

Thakthelackal Varunak

Date



RECORDED: 11/12/2004

PATENT  
REEL: 016366 FRAME: 0278

**TRADUCCIÓN DEL INGLÉS AL ESPAÑOL DE CINCO PÁGINAS DE UN DOCUMENTO ORIGINAL ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA MERYAM ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-8917116; MAYO 16 DE 2005**

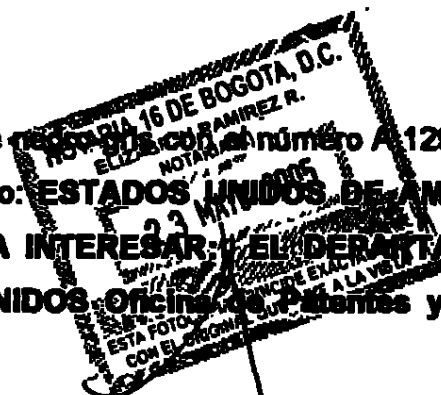
**"APOSTILLE**

**(Convención De La Haya de octubre 5 de 1961)**

1. País: ESTADOS UNIDOS DE AMÉRICA
  2. Este documento público fue firmado por: M.K. HAWKINS
  3. En su calidad de: FUNCIONARIO DE CERTIFICACIONES
  4. Lleva el sello de: LA OFICINA DE PATENTES Y MARCAS REGISTRADAS
- CERTIFICADO**
5. En: WASHINGTON, D.C.
  6. El día: veintidós de marzo de 2005
  7. Por: ASISTENTE DEL FUNCIONARIO DE AUTENTICACIONES DEL DEPARTAMENTO DE ESTADO DE ESTADOS UNIDOS
  8. No. 05014224-4
  9. Sello: SELLO SECO DEL DEPARTAMENTO DE ESTADO DE ESTADOS UNIDOS
  10. Firma: SONYA N. JOHNSON

**SEGUNDA PÁGINA:**

Emitida en papelería especial con reborde negro en la parte superior izquierda del documento: **ESTADOS UNIDOS DE AMÉRICA - PARA AQUÉLLOS A QUIENES PUEDA INTERESAR: EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS Oficina de Patentes y Marcas Registradas de los Estados Unidos**



18 de marzo de 2005

*Meryam Adam*

Traductora e Intérprete Oficial  
Inglés-Español / Español-Inglés,  
Según Resolución 0526 de 11 de Junio de 2002  
Ente del Ministerio de Justicia y del Derecho  
**Meryam Adam J.**  
CC 92 452 413 064

MINISTERIO DE RELACIONES  
EXTERIORES

NO SE ASUME  
RESPONSABILIDAD DEL TEXTO

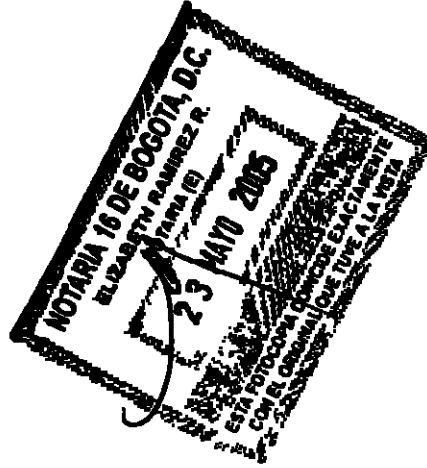
112530

2005 MAY 19 P 1:26

*Heriberto Adam*  
CUYA FIRMA APARECE EN EL  
DOCUMENTO DESEMPEÑA  
LAS FUNCIONES INDICADAS

*sec*

JEFE DE LEGALIZACIONES  
SANTAFE DE BOGOTA S.C.



**ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS SON COPIA FIEL DEL DOCUMENTO QUE FUE REGISTRADO EN ESTA OFICINA EL DÍA 12 DE NOVIEMBRE DE 2004.**

**Estampilla dorada del Departamento de Comercio - Oficina de Patentes y Marcas Registradas de los Estados Unidos**

**Bajo la potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS**

**(Firma)**  
**M.K. HAWKINS**  
**Funcionario de Certificaciones**

**TERCERA PÁGINA:** Papel blanco con el número CH \$40.00 10882134 al margen derecho

**CESIÓN DE LA PATENTE**

**Versión Electrónica v1.1**

**Versión Impresa v1.1**

|                                            |                     |
|--------------------------------------------|---------------------|
| <b>Tipo de Comisión:</b>                   | <b>Nueva Cesión</b> |
| <b>Característica de la Transferencia:</b> | <b>Cesión</b>       |

**Información de La Parte que Transfiere**

| <b>Nombre</b>               | <b>Fecha de Ejecución</b> |
|-----------------------------|---------------------------|
| <b>Bryan Mark Willman</b>   | <b>11/02/04</b>           |
| <b>Pierre Jacomet</b>       | <b>11/02/04</b>           |
| <b>Derek Norman LaSalle</b> | <b>11/04/04</b>           |



*M*

**Información de la Parte Receptora**

|                   |                              |
|-------------------|------------------------------|
| <b>Nombre:</b>    | <b>MICROSOFT CORPORATION</b> |
| <b>Dirección:</b> | <b>One Microsoft Way</b>     |

133

|                |            |
|----------------|------------|
| Ciudad:        | Redmond    |
| Estado/País:   | WASHINGTON |
| Código Postal: | 98052      |

Número Total de Propiedades: 1

|                      |          |
|----------------------|----------|
| Tipo de Propiedad:   | Número   |
| Número de Solicitud: | 10882134 |

### CORRESPONDENCIA

Número de telefax: (215) 568-3439

*La correspondencia será enviada por correo de los Estados Unidos si el envío por telefax falla*

Teléfono: 215-568-3100

Correo Electrónico: [pillero@woodcock.com](mailto:pillero@woodcock.com)

Nombre del Correspondiente: Woodcock Washburn LLP

Renglón 1 de la Dirección: One Liberty Place, 46th. floor

Renglón 4 de la Dirección: Philadelphia, PENNSYLVANIA 19103

|                             |                          |
|-----------------------------|--------------------------|
| NOMBRE DE QUIEN ENVÍA:      | Charlotte Pillero-Morris |
| Número de Anexos: 10        |                          |
| Fuente = 10882134#page1.tif |                          |
| Fuente = 10882134#page2.tif |                          |
| Fuente = 10882134#page3.tif |                          |
| Fuente = 10882134#page4.tif |                          |

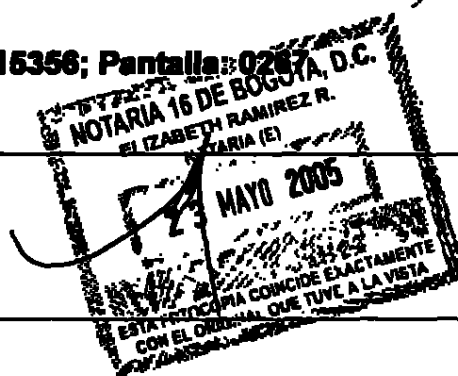
500011858

PATENTE

Carreta: 015356; Pantalla: 0267

### CUARTA PÁGINA

|                             |
|-----------------------------|
| Fuente = 10882134#page5.tif |
| Fuente = 10882134#page6.tif |
| Fuente = 10882134#page7.tif |



my

Fuente = 10882134#page8.tif  
Fuente = 10882134#page9.tif  
Fuente = 10882134#page10.tif

**PATENTE**

**Carrete: 015356; Pantalla: 0268**

**QUINTA PÁGINA: Con encabezado de envío de fax Nov 08 2004 17:42 FR  
Microsoft Building 8 Para 912155683439 P02/06**

**EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS  
ESTADOS UNIDOS**

**Número de Serie....Aún sin asignar**

**Fecha de Registro....La presente**

**Autoría del Invento...Bryan Mark Willman et al.**

**Parte Solicitante...Microsoft Corporation**

**Expediente del Abogado No.....MSFT-3013/305832.01**

**Título... SISTEMA Y MÉTODO PARA INICIO DE SISTEMA OPERATIVO  
PROTEGIDO UTILIZANDO VALIDACIÓN DE LA CONDICIÓN**

**CESIÓN DE LA PATENTE**

**PARTES DE LA CESIÓN**

**Cedente (s):**

**Bryan Mark Willman  
10901 – 80th Place NE  
Kirkland, Washington 98034**

**Paul England  
16659 Northup Way  
Bellevue, Washington 98008**



M

185

**Kenneth D. Ray**  
**920 North 34th Street # 88**  
**Seattle, Washington 98103**

**Jamie Hunter**  
**14226 93rd Ave NE**  
**Bothell, Washington 98011**

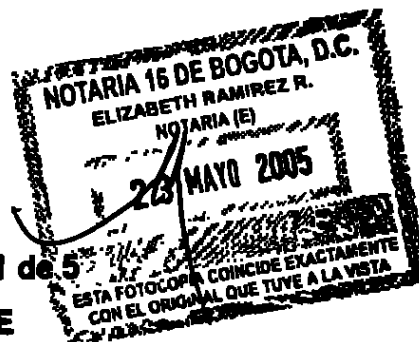
**Lonny Dean McMichael**  
**9950 229th Ln NE**  
**Redmond, Washington 98053**

**Derek Norman LaSalle**  
**5509 162nd Ave NE**  
**Redmond, Washington 98052**

**Pierre Jacomet**  
**1221 240th Way SE**  
**Sammamish, Washington 98075**

**Mark Eliot Paley**  
**18112 NE 31st Street**  
**Redmond, Washington 98052**

**Thekkthalackal Varugis Kuriel**  
**22120 NE 23rd Street**  
**Sammamish, Washington 98075**



m

186

Carrete: 015356; Pantalla: 0269

**SEXTA PÁGINA:** Con encabezado de envío de fax Nov 08 2004 17:42 FR  
Microsoft Building 8 Para 912155683439 P03/06

David B. Cross  
27524 NE 31st Court  
Redmond Washington 98053  
Página 2 de 5

**PATENTE**

Carrete: 015356; Pantalla: 0270

**SÉPTIMA PÁGINA:** Con encabezado de envío de fax Nov 08 2004 17:42 FR  
Microsoft Building 8 Para 912155683439 P04/06

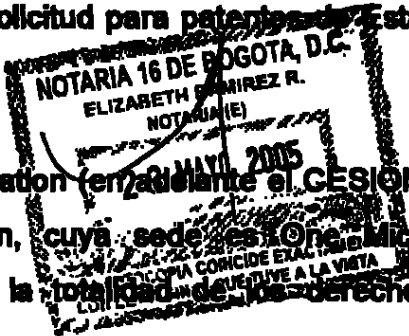
**Cesionario:**

Microsoft Corporation  
Corporation in the State of Washington  
One Microsoft Way  
Redmond, WA 98052

**CONVENIO**

CONSIDERANDO QUE EL (LOS) CEDENTE (S) (mencionados anteriormente) son el (los) creador (es) de un invento de acuerdo con lo descrito y reivindicado en la especificación que hace parte de la solicitud para patentes de Estados Unidos arriba indicados;

CONSIDERANDO, que Microsoft Corporation (en adelante el CESIONARIO), una corporación del estado de Washington, cuya sede es One Microsoft Way, Redmond, WA 98052, intenta adquirir la totalidad de los derechos, títulos e



m

intereses en el invento y por éste y en o por cualquier patente que pueda serle otorgado en los Estados Unidos y en cualquiera y todos los países extranjeros;

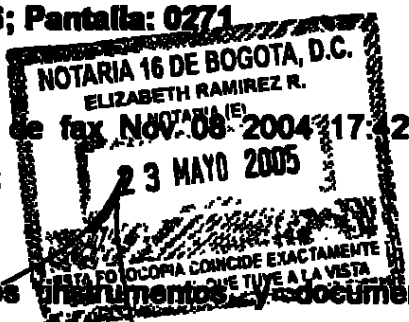
AHORA, POR LO TANTO, a cambio de contraprestaciones onerosas, el recibo de las cuales se reconocen en este documento, los CEDENTES por el presente venden, ceden y transfieren al CESIONARIO la totalidad de los derechos, títulos e intereses en y por dicho invento, solicitud y cualesquiera y todas las patentes que puedan ser otorgadas sobre el mismo en los Estados Unidos de América y sus posesiones territoriales, así como en todos y cualesquiera países extranjeros, y en todos y cualesquiera divisiones, re-emisiones y extensiones de la misma, incluyendo el derecho a registrar solicitudes en el extranjero directamente a nombre del CESIONARIO y a reclamar derechos prioritarios derivados de dicha solicitud en los Estados Unidos, a los cuales tienen derecho dichas solicitudes extranjeras en virtud de convenios internacionales, tratados u otras maneras; dicho invento, solicitud, y todas las patentes sobre el mismo serán conservados y en posesión del CESIONARIO y sus sucesores y apoderados, para su uso y beneficio y el de sus sucesores y apoderados, tan completa y totalmente como si fuesen conservados y en posesión de los CEDENTES, si esta cesión, transferencia o venta no se hubiera realizado. Los CEDENTES por el presente autorizan y solicitan al Comisionado de Patentes y Marcas Registradas emitir todas las patentes sobre dicho invento al CESIONARIO.

Página 3 de 5

PATENTE

Carrete: 015356; Pantalla: 0271

OCTAVA PÁGINA: Con encabezado de envío de fax Nov. 08 2004 17:42 FR  
Microsoft Building 8 Para 912155683439 P05/06



Los CEDENTES acuerdan ejecutar todos los ~~instrumentos~~ y documentos requeridos para llevar a cabo y proseguir la solicitud de las patentes en los Estados Unidos y en el extranjero sobre dicho invento para fines de litigios

relacionados con los mismos, o para proteger dicho título sobre el invento o patentes.

Página 4 de 5

PATENTE

Carrete: 015356; Pantalla: 0272

NOVENA PÁGINA: Con encabezado de envío de fax Nov 08 2004 17:42 FR  
Microsoft Building 8 Para 912155683439 P06/06

2004- NOV - 02

F cha

(Firma)

Bryan Mark Wilman

(En blanco)

Fecha

(En blanco)

Paul England

(En blanco)

Fecha

(En blanco)

Kenneth D. Ray

(En blanco)

Fecha

(En blanco)

Jamie Hunter

(En blanco)

Fecha

(En blanco)

Lonny Dean McMichael

(En blanco)

Fecha

(En blanco)

Derek Norman LaSalle

2004- 11 - 02

Fecha

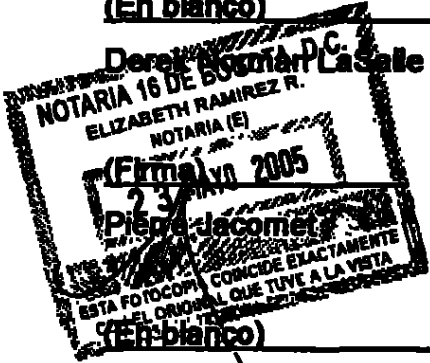
(Firma)

23 Mayo 2005

Pierre Jacomier

(En blanco)

(En blanco)



mf

|                    |                               |
|--------------------|-------------------------------|
| Fecha              | Mark Eliot Paley              |
| <u>(En blanco)</u> | <u>(En blanco)</u>            |
| Fecha              | Thekkthalackal Varugis Kurien |
| <u>(En blanco)</u> | <u>(En blanco)</u>            |
| Fecha              | David B. Cross                |

Página 4 de 5                      **PATENTE**  
Carrete: 015356; Pantalla: 0273

**Total Páginas 06**  
**DÉCIMA PÁGINA:** Con encabezado de envío de fax Nov 05 2004 17:59 FR  
Microsoft 425 703 2836 Para 912155683439 P02/06

**EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS  
ESTADOS UNIDOS**

Número de Serie.....Aún sin asignar  
Fecha de Registro....La presente  
Autoría del Invento...Bryan Mark Willman et al.  
Parte Solicitante...Microsoft Corporation  
Expediente del Abogado No.....MSFT-3013/305832.01  
Título... SISTEMA Y MÉTODO PARA INICIO DE SISTEMA OPERATIVO  
PROTEGIDO UTILIZANDO VALIDACIÓN DE LA CONDICIÓN

**CESIÓN DE LA PATENTE**

**PARTES DE LA CESIÓN**

**Cedente (s):**

Bryan Mark Willman  
10901 – 80th Place NE



*Handwritten mark*

**Kirkland, Washington 98034**

**Paul England**

**16659 Northup Way**

**Bellevue, Washington 98008**

**Kenneth D. Ray**

**920 North 34th Street # 88**

**Seattle, Washington 98103**

**Jamie Hunter**

**14226 93rd Ave NE**

**Bothell, Washington 98011**

**Lonny Dean McMichael**

**9950 229th Ln NE**

**Redmond, Washington 98053**

**Derek Norman LaSalle**

**5509 162nd Ave NE**

**REdmond, Washington 98052**

**Pierre Jacomet**

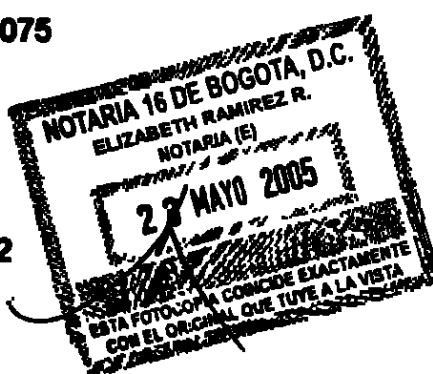
**1221 240th Way SE**

**Sammamish, Washington 98075**

**Mark Eliot Paley**

**18112 NE 31st Street**

**Redmond, Washington 98052**



m

Thekkthalackal Varugis Kurien  
22120 NE 23rd Street  
Sammamish, Washington 98075

UNDÉCIMA PÁGINA: Con encabezado de envío de fax Nov 05 2004 17:59 FR  
Microsoft 425 703 2836 Para 912155683439 P03/06

David B. Cross  
27524 NE 31st Court  
Redmond Washington 98053

**PATENTE**

Carrete: 015356; Pantalla: 0276

DÚODÉCIMA PÁGINA: Con encabezado de envío de fax Nov 05 2004 17:59 FR  
Microsoft 425 703 2836 Para 912155683439 P04/06

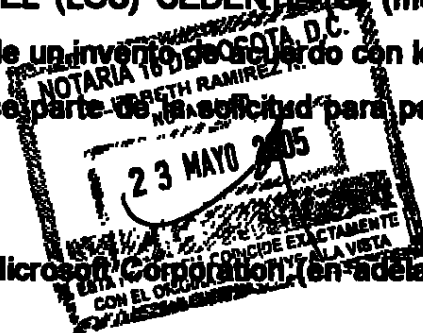
**Cesionario:**

Microsoft Corporation  
Corporation in the State of Washington  
One Microsoft Way  
Redmond, WA 98052

**CONVENIO**

CONSIDERANDO QUE EL (LOS) CEDENTE (S) (mencionados anteriormente) son el (los) creador (es) de un invento de acuerdo con lo descrito y reivindicado en la especificación que hace parte de la solicitud para patentes de Estados Unidos arriba indicados;

CONSIDERANDO, que Microsoft Corporation (en adelante el CESIONARIO), una



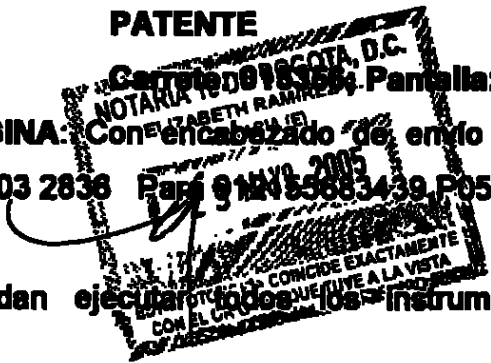
m

corporación del estado de Washington, cuya sede es One Microsoft Way, Redmond, WA 98052, intenta adquirir la totalidad de los derechos, títulos e intereses en el invento y por éste y en o por cualquier patente que pueda serle otorgado en los Estados Unidos y en cualquiera y todos los países extranjeros;

AHORA, POR LO TANTO, a cambio de contraprestaciones onerosas, el recibo de las cuales se reconocen en este documento, los CEDENTES por el presente venden, ceden y transfieren al CESIONARIO la totalidad de los derechos, títulos e intereses en y por dicho invento, solicitud y cualesquiera y todas las patentes que puedan ser otorgadas sobre el mismo en los Estados Unidos de América y sus posesiones territoriales, así como en todos y cualesquiera países extranjeros, y en todos y cualesquiera divisiones, re-emisiones y extensiones de la misma, incluyendo el derecho a registrar solicitudes en el extranjero directamente a nombre del CESIONARIO y a reclamar derechos prioritarios derivados de dicha solicitud en los Estados Unidos, a los cuales tienen derecho dichas solicitudes extranjeras en virtud de convenios internacionales, tratados u otras maneras; dicho invento, solicitud, y todas las patentes sobre el mismo serán conservados y en posesión del CESIONARIO y sus sucesores y apoderados, para su uso y beneficio y el de sus sucesores y apoderados, tan completa y totalmente como si fuesen conservados y en posesión de los CEDENTES, si esta cesión, transferencia o venta no se hubiera realizado. Los CEDENTES por el presente autorizan y solicitan al Comisionado de Patentes y Marcas Registradas emitir todas las patentes sobre dicho invento al CESIONARIO.

**PATENTE**  
Carpeten 0183554 Pantalla: 0276  
DÉCIMATERCERA PÁGINA: Con encabezado de envío de fax Nov 05 2004  
17:59 FR Microsoft 425 703 2836 Page 9 of 11 55663439 P05/06

Los CEDENTES acuerdan ejecutar todos los instrumentos y documentos



my

requeridos para llevar a cabo y proseguir la solicitud de las patentes en los Estados Unidos y en el extranjero sobre dicho invento para fines de litigios relacionados con los mismos, o para proteger dicho título sobre el invento o patentes.

PATENTE

Carrete: 015356; Pantalla: 0277

DÉCIMACUARTA PÁGINA: Con encabezado de envío de fax Nov 05 2004 17:59  
FR Microsoft 425 703 2836 Para 912155683439 P06/06

(En blanco)  
Fecha

(En blanco)  
Bryan Mark Willman

(En blanco)  
Fecha

(En blanco)  
Paul England

(En blanco)  
Fecha

(En blanco)  
Kenneth D. Ray

(En blanco)  
Fecha

(En blanco)  
Jamie Hunter

(En blanco)  
Fecha

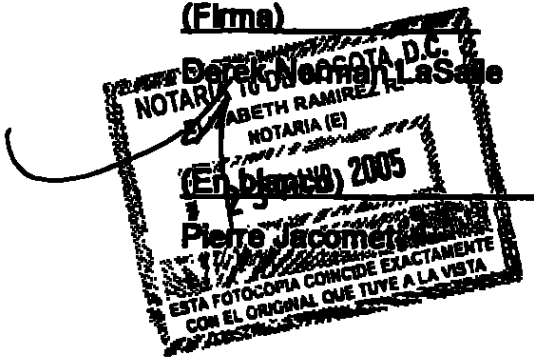
(En blanco)  
Lonny Dean McMichael

Nov 4, 2004  
Fecha

(Firma)  
Derek Newman LaSalle

(En blanco)  
Fecha

(En blanco) 2005  
Pierre Jacomet



07

(En blanco)

Fecha

(En blanco)

Mark Eliot Paley

(En blanco)

Fecha

(En blanco)

Thekkthalackal Varugis Kurien

(En blanco)

Fecha

(En blanco)

David B. Cross

Registrado: 11/12/2004

**PATENTE**

**Carreta: 015356; Pantalla: 0278**

**CONTRAPORTADA: LOGO DEL DEPARTAMENTO DE COMERCIO DE  
ESTADOS UNIDOS - OFICINA DE PATENTES Y MARCAS REGISTRADAS**

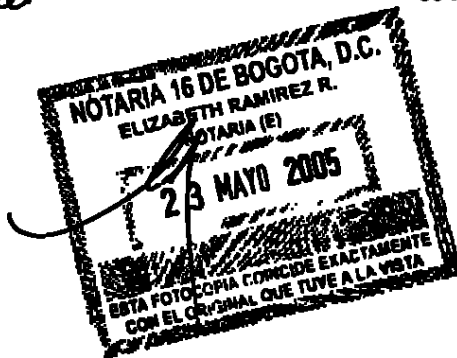
CERTIFICO QUE ESTA ES UNA TRADUCCION FIEL DEL INGLES AL  
ESPAÑOL DE UN DOCUMENTO QUE HE TENIDO A LA VISTA; QUE SOY  
TRADUCTORA E INTERPRETE OFICIAL ACREDITADA POR RESOLUCIONES  
0036 Y 526 DE LA UNIVERSIDAD NACIONAL DE COLOMBIA Y DEL  
MINISTERIO DE LA JUSTICIA RESPECTIVAMENTE, Y QUE ESTOY  
REGISTRADA ANTE EL MINISTERIO DE RELACIONES EXTERIORES DE  
COLOMBIA

Bogotá, mayo 16 de 2005

*Meryam Adam J.*

MERYAM ADAM J.  
C.C. 52.252.613

Traductora e interprete Oficial  
Inglés-Español / Español-Inglés,  
Según Resolución 0586 de 11 de Junio de 2003  
emitida por el Ministerio de Justicia y del Derecho  
**Meryam Adam J.**  
C.C. 52.252.613



**APOSTILLE**

*(Convention de La Haye du 5 octobre 1961)*

1. Country: United States of America

This public document

2. has been signed by T. Lawrence

3. acting in the capacity of Certifying Officer

4. bears the seal/stamp of Patent and Trademark Office

**Certified**

5. at Washington, D.C.

6. the thirtieth of March, 2005

7. by Authentication Officer, United States Department of State

8. No. 05015050-6

9. Seal/Stamp:

10. Signature:

*Denitra T. Hawkins*

Denitra T. Hawkins



A 1300128

# THE UNITED STATES OF AMERICA

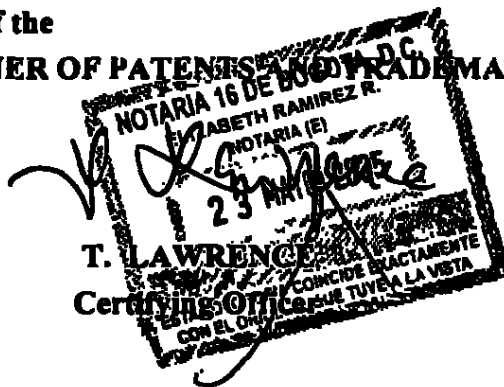
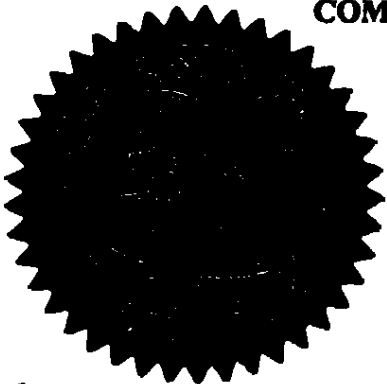
TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office

March 28, 2005

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE  
RECORDS OF THIS OFFICE OF A DOCUMENT RECORDED ON  
OCTOBER 25, 2004.

By Authority of the  
COMMISSIONER OF PATENTS AND TRADEMARKS



197

PATENT ASSIGNMENT

Version v1.1  
at Version v1.1

|                     |                |
|---------------------|----------------|
| ASSIGN TYPE:        | NEW ASSIGNMENT |
| TYPE OF CONVEYANCE: | ASSIGNMENT     |

DEEDING PARTY DATA

| Name                         | Execution Date |
|------------------------------|----------------|
| England                      | 08/29/2004     |
| Ernest D. Ray                | 08/29/2004     |
| He Hunter                    | 08/29/2004     |
| Henry Dean McMichael         | 08/29/2004     |
| Mark Elliot Paley            | 08/29/2004     |
| Radhakrishnan Varugis Kurian | 08/29/2004     |
| David B. Cross               | 08/29/2004     |

RECEIVING PARTY DATA

|                 |                       |
|-----------------|-----------------------|
| Name:           | Microsoft Corporation |
| Street Address: | One Microsoft Way     |
| City:           | Redmond               |
| State/Country:  | WASHINGTON            |
| Postal Code:    | 98052                 |

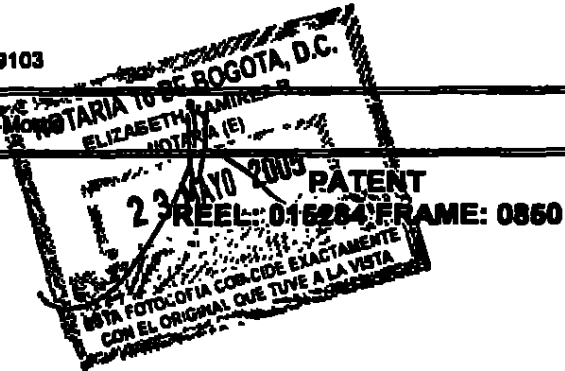
PROPERTY NUMBERS Total: 1

| Property Type       | Number   |
|---------------------|----------|
| Application Number: | 10682134 |

CORRESPONDENCE DATA

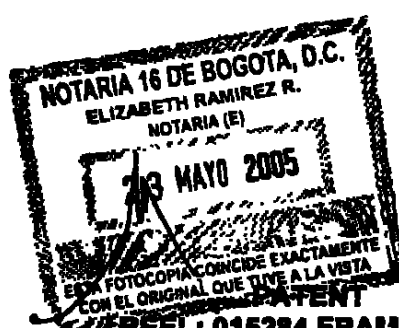
Fax Number: (216)586-3439  
*Correspondence will be sent via US Mail when the fax attempt is unsuccessful.*  
Phone: 2152583100  
Email: pillero@woodcock.com  
Correspondent Name: Woodcock Washburn LLP  
Address Line 1: One Liberty Place, 48th floor  
Address Line 4: Philadelphia, PENNSYLVANIA 19103

NAME OF SUBMITTER: Charlotte Pillero-Morales



CH \$40.00 10682134

documents: 5  
 10882134#page1.tif  
 10882134#page2.tif  
 10882134#page3.tif  
 10882134#page4.tif  
 10882134#page5.tif



REEL: 015284 FRAME: 0851

# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number \_\_\_\_\_ Not Yet Assigned  
 Filing Date \_\_\_\_\_ Harnwell  
 Invention \_\_\_\_\_ Bryan Mark Wilman, et al.  
 Applicant \_\_\_\_\_ Microsoft Corporation  
 Attorney's Docket No. \_\_\_\_\_ MEFT-3013/905832.01  
 Title: **SYSTEM AND METHOD FOR PROTECTED OPERATING SYSTEM BOOT  
 USING STATE VALIDATION**

## PATENT ASSIGNMENT

### PARTIES TO THE ASSIGNMENT

#### Assignor(s):

Bryan Mark Wilman  
 10901 - 80th Place, NE  
 Kirkland, Washington 98034

Paul England  
 16699 Northrup Way  
 Bellevue, Washington 98008

Kenneth D. Ray  
 920 North 34th Street, #68  
 Seattle, Washington 98105

Jamie Hunter  
 14226 83rd Ave NE  
 Bothell, WA 98011

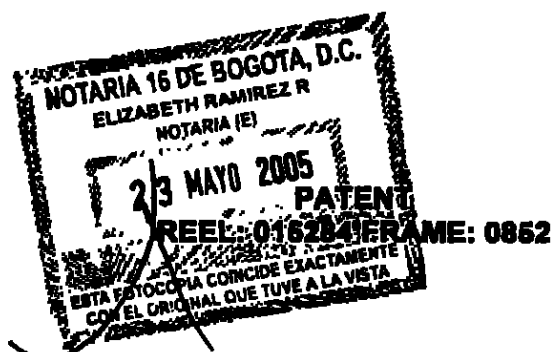
Lenny Dean McMichael  
 9950 229th Ln NE  
 Redmond, WA 98053

Deak Norman LaFalle  
 8809 162nd Ave. NE  
 Redmond, WA 98052

Pierre Jacquinet  
 1221 240th Way, SE  
 Sammamish, Washington 98075

Mark Eliot Paley  
 18112 NE 31st Street  
 Redmond, WA 98052

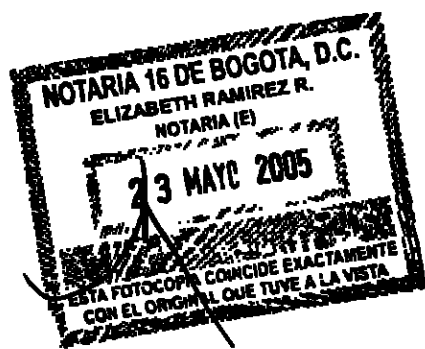
Thekkilackal Varughese Kurian  
 22120 NE 23rd Street  
 Sammamish, Washington 98074



200

JUN 30 2004 18:38 FR MICROSOFT BLD 48 425 936 7329 TO 912153683439 P.03/09

David E. Cross  
27224 NE 51st Court  
Redmond, Washington 98053



PATENT  
REEL: 016284 FRAME: 0863

**Assignor**

Microsoft Corporation  
 Corporation in the State of Washington  
 One Microsoft Way  
 Redmond, WA 98052

**AGREEMENT**

WHEREAS, ASSIGNOR(S) (listed above) are inventor(s) of an invention entitled as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNOR(S) hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, renewals and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR(S) had this assignment, transfer and sale not been made.

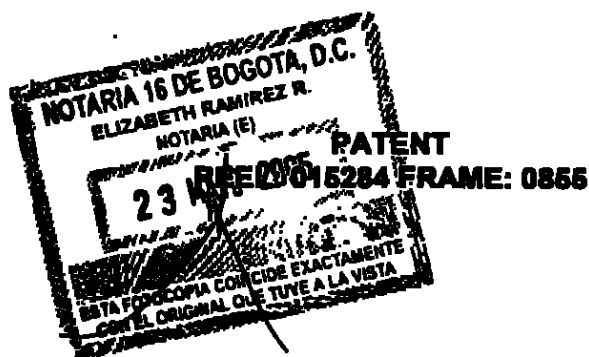
ASSIGNOR(S) hereby authorize and request the Commissioner of Patents and Trademarks to issue a patent on said invention to ASSIGNEE. ASSIGNOR(S) agree to execute all documents and documents



FRAME: 0854

252

quired for the making and prosecution of applications for United States and foreign letters  
patent on said invention, for litigation regarding said letters patent, or for the purpose of  
protecting title to said invention or letters patent therefor.



253

Date

6/29/06

Date

6/30/04

Date

6/29/04

Date

6-29-04

Date

Date

Date

6/29/04

Date

6/29/04

Date

6-29-04

Date

Bryan Mark Wilman

Paul England

Kenneth D. Ray

Janis Hunter

Lenny Dean McMichael

Derek Norman LaSalle

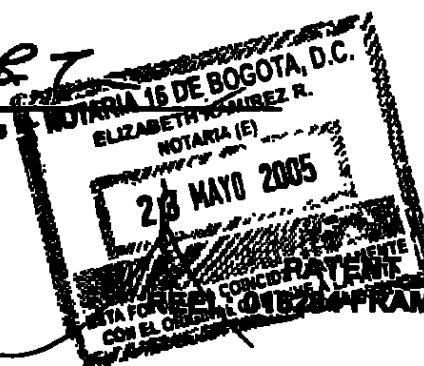
Pierre Jacomet

Mark Elliot Foley

Thakthalaalai Varugis Kurian

David E. Cross

RECORDED: 10/25/2004



FRAME: 0866

204

**TRADUCCIÓN DEL INGLÉS AL ESPAÑOL DE NUEVE PÁGINAS DE UN DOCUMENTO ORIGINAL ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA MERYAM ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-8917116; MAYO 16 DE 2005**

**"APOSTILLE**

**(Convención De La Haya de octubre 5 de 1961)**

1. País: ESTADOS UNIDOS DE AMÉRICA
2. Este documento público fue firmado por: T. LAWRENCE
3. En su calidad de: FUNCIONARIO DE CERTIFICACIONES
4. Lleva el sello de: LA OFICINA DE PATENTES Y MARCAS REGISTRADAS  
**CERTIFICADO**
5. En: WASHINGTON, D.C.
6. El día: treinta de marzo de 2005
7. Por: ASISTENTE DEL FUNCIONARIO DE AUTENTICACIONES DEL DEPARTAMENTO DE ESTADO DE ESTADOS UNIDOS
8. No. 05015050-6
9. Sello: SELLO SECO DEL DEPARTAMENTO DE ESTADO DE ESTADOS UNIDOS
10. Firma: DENITRA T. HAWKINS

**SEGUNDA PÁGINA:**

Emitida en papelería especial con reborde negro-gris con el número A 1300135 en la parte superior izquierda del documento: **ESTADOS UNIDOS DE AMÉRICA - PARA AQUÉLLOS A QUIENES PUEDA INTERESAR EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS Oficina de Patentes y Marcas Registradas de los Estados Unidos**

*Meryam Adam*

Traductora e Intérprete Oficial  
Inglés-Español / Español-Inglés  
Según Resolución 0526 de 11 de Junio de 2004 EL ORIGINAL QUE TUVE A LA VISTA  
emitida por el Ministerio de Justicia y del Poder Judicial  
**Meryam Adam J.**  
CC 98 258 613 864

NOTARIA 104 DE WASHINGTON, D.C.  
ELIZABETH RAMIREZ  
23 MAYO 2005  
28 de marzo de 2005  
ESTA FOTOCOPIA COINCIDE EXACTAMENTE

205

**ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS SON COPIA FIEL DEL DOCUMENTO QUE FUE REGISTRADO EN ESTA OFICINA EL DÍA 25 DE OCTUBRE DE 2004.**

**Estampilla dorada del Departamento de Comercio - Oficina de Patentes y Marcas Registradas de los Estados Unidos**

**Bajo la potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS**

**(Firma)**  
**T. LAWRENCE**  
**Funcionario de Certificaciones**

**TERCERA PÁGINA:** Papel blanco con el número CH \$40.00 10882134 al margen derecho

**CESIÓN DE LA PATENTE**

**Versión Electrónica v1.1**

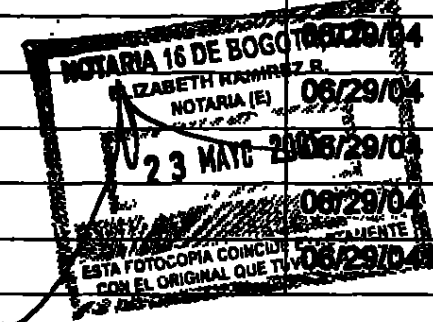
**Versión Impresa v1.1**

|                                            |                     |
|--------------------------------------------|---------------------|
| <b>Tipo de Comisión:</b>                   | <b>Nueva Cesión</b> |
| <b>Característica de la Transferencia:</b> | <b>Cesión</b>       |

M

**Información de La Parte que Transfiere**

| <b>Nombre</b>                        | <b>Fecha de Ejecución</b> |
|--------------------------------------|---------------------------|
| <b>Paul England</b>                  | <b>06/29/04</b>           |
| <b>Kenneth D. Ray</b>                | <b>06/30/04</b>           |
| <b>Jamie Hunter</b>                  | <b>06/29/04</b>           |
| <b>Lonny Dean McMichael</b>          | <b>06/29/04</b>           |
| <b>Mark Eliot Paley</b>              | <b>06/29/04</b>           |
| <b>Thekkthalackal Varugis Kurlen</b> | <b>06/29/04</b>           |
| <b>David B. Cross</b>                | <b>06/29/04</b>           |



206

#### Información de la Parte Receptora

|                |                       |
|----------------|-----------------------|
| Nombre:        | MICROSOFT CORPORATION |
| Dirección:     | One Microsoft Way     |
| Ciudad:        | Redmond               |
| Estado/País:   | WASHINGTON            |
| Código Postal: | 98052                 |

Número Total de Propiedades: 1

|                      |          |
|----------------------|----------|
| Tipo de Propiedad:   | Número   |
| Número de Solicitud: | 10882134 |

#### CORRESPONDENCIA

Número de telefax: (215) 588-3439

*La correspondencia será enviada por correo de los Estados Unidos si el envío por telefax falla*

Teléfono: 215-258-3100

Correo Electrónico: [piliro@woodcock.com](mailto:piliro@woodcock.com)

Nombre del Correspondiente: Woodcock Washburn LLP

Renglón 1 de la Dirección: One Liberty Place, 48th. floor

Renglón 4 de la Dirección: Philadelphia, PENNSYLVANIA 19103

msi  
D

|                        |                         |
|------------------------|-------------------------|
| NOMBRE DE QUIEN ENVÍA: | Charlotte Piliro-Morris |
|------------------------|-------------------------|

#### PATENTE

Carrete: 015284; Pantalla: 0850

#### CUARTA PÁGINA

Número de Anexos: 5

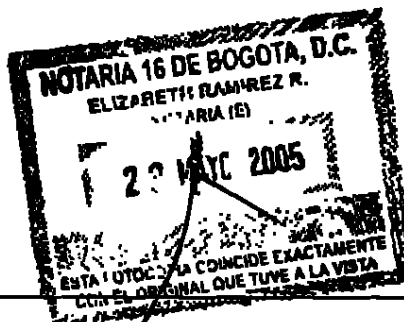
Fuente = 10882134#page1.tif

Fuente = 10882134#page2.tif

Fuente = 10882134#page3.tif

Fuente = 10882134#page4.tif

Fuente = 10882134#page5.tif



**PATENTE**

**Carrete: 016284; Pantalla: 0851**

**QUINTA PÁGINA: Con encabezado de envío de fax Jun 30 2004 10:29 FR  
Microsoft Building 40 425 936 7329 Para 912155683439 P02/09**

**EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS  
ESTADOS UNIDOS**

**Número de Serie....Aún sin asignar**

**Fecha de Registro....La presente**

**Autoría del Invento...Bryan Mark Willman et al.**

**Parte Solicitante...Microsoft Corporation**

**Expediente del Abogado No.....MSFT-3013/305832.01**

**Título... SISTEMA Y MÉTODO PARA INICIO DE SISTEMA OPERATIVO  
PROTEGIDO UTILIZANDO VALIDACIÓN DE LA CONDICIÓN**

**CESIÓN DE LA PATENTE**

**PARTES DE LA CESIÓN**

**Cedente (s):**

**Bryan Mark Willman**

**10901 – 80th Place NE**

**Kirkland, Washington 98034**

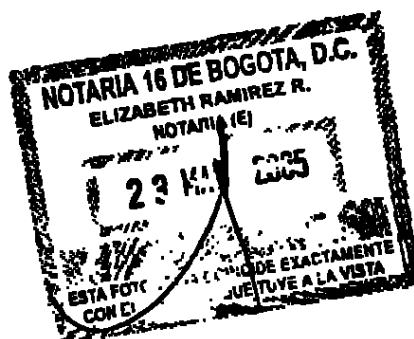
**Paul England**

**16659 Northup Way**

**Bellevue, Washington 98008**

**Kenneth D. Ray**

**920 North 34th Street # 88**



208

Seattle, Washington 98103

Jamie Hunter

14226 93rd Ave NE

Bothell, Washington 98011

Lonny Dean McMichael

9950 229th Ln NE

Redmond, Washington 98053

Derek Norman LaSalle

5509 162nd Ave NE

Redmond, Washington 98052

Pierre Jacomet

1221 240th Way SE

Sammamish, Washington 98075

Mark Eliot Paley

18112 NE 31st Street

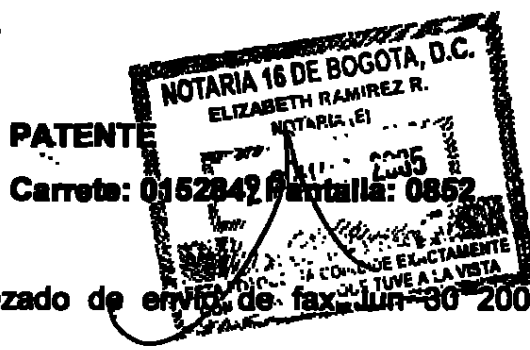
Redmond, Washington 98052

any

Thekkthalackal Varugis Kurien

22120 NE 23rd Street

Sammamish, Washington 98074



SEXTA PÁGINA: Con encabezado de envío de fax JUN 30 2004 10:29 FR

209

Microsoft Building 40 425 936 7329 Para 912155683439 P03/09

David B. Cross

27524 NE 31st Court

Redmond Washington 98053

**PATENTE**

**Carrete: 015284; Pantalla: 0853**

**SÉPTIMA PÁGINA:** Con encabezado de envío de fax Jun 30 2004 10:29 FR  
Microsoft Building 40 425 936 7329 Para 912155683439 P04/09

**Cesionario:**

Microsoft Corporation

Corporation in the State of Washington

One Microsoft Way

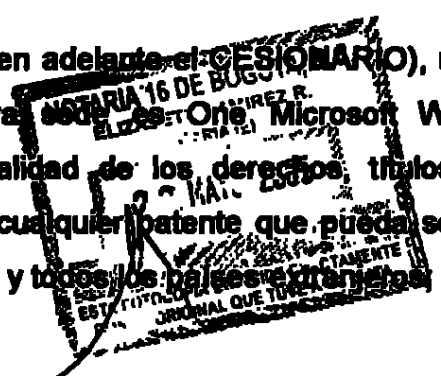
Redmond, WA 98052

**CONVENIO**

**CONSIDERANDO QUE EL (LOS) CEDENTE (S) (mencionados anteriormente) son el (los) creador (es) de un invento de acuerdo con lo descrito y reivindicado en la especificación que hace parte de la solicitud para patentes de Estados Unidos arriba indicados;**

*mg*

**CONSIDERANDO, que Microsoft Corporation (en adelante el ~~CEDENTE~~ **CESIONARIO**), una corporación del estado de Washington, cuya sede es One Microsoft Way, Redmond, WA 98052, intenta adquirir la totalidad de los derechos, títulos e intereses en el invento y por éste y en o por cualquier patente que pueda serle otorgado en los Estados Unidos y en cualquiera y todos los países extranjeros;**



210

AHORA, POR LO TANTO, a cambio de contraprestaciones onerosas, el recibo de las cuales se reconocen en este documento, los CEDENTES por el presente venden, ceden y transfieren al CESIONARIO la totalidad de los derechos, títulos e intereses en y por dicho invento, solicitud y cualesquiera y todas las patentes que puedan ser otorgadas sobre el mismo en los Estados Unidos de América y sus posesiones territoriales, así como en todos y cualesquiera países extranjeros, y en todos y cualesquiera divisiones, re-emisiones y extensiones de la misma, incluyendo el derecho a registrar solicitudes en el extranjero directamente a nombre del CESIONARIO y a reclamar derechos prioritarios derivados de dicha solicitud en los Estados Unidos, a los cuales tienen derecho dichas solicitudes extranjeras en virtud de convenios internacionales, tratados u otras maneras; dicho invento, solicitud, y todas las patentes sobre el mismo serán conservados y en posesión del CESIONARIO y sus sucesores y apoderados, para su uso y beneficio y el de sus sucesores y apoderados, tan completa y totalmente como si fuesen conservados y en posesión de los CEDENTES, si esta cesión, transferencia o venta no se hubiera realizado. Los CEDENTES por el presente autorizan y solicitan al Comisionado de Patentes y Marcas Registradas emitir todas las patentes sobre dicho invento al CESIONARIO.

**PATENTE**

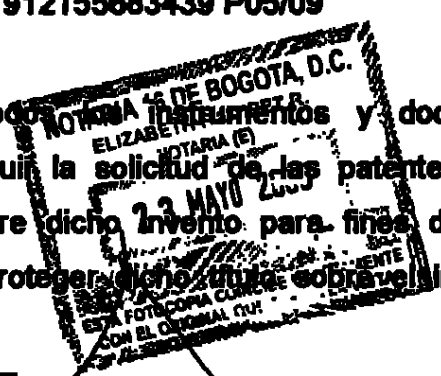
**Carrete: 015284; Pantalla: 0854**

M

**OCTAVA PÁGINA:** Con encabezado de envío de fax Jun 30 2004 10:29 FR  
Microsoft Building 40 425 936 7329 Para 912155683439 P05/09

Los CEDENTES acuerdan ejecutar todos los instrumentos y documentos requeridos para llevar a cabo y proseguir la solicitud de las patentes en los Estados Unidos y en el extranjero sobre dicho invento para fines de litigios relacionados con los mismos, o para proteger dicho título sobre el invento o patentes.

**PATENTE**



Carrete: 015284; Pantalla: 0855

NOVENA PÁGINA: Con encabezado de envío de fax Jun 30 2004 10:29 FR  
Microsoft Building 40 425 936 7329 Para 912155683439 P06/09

(En blanco)  
Fecha

(En blanco)  
Bryan Mark Willman

6/29/(número ilegible)  
Fecha

(Firma)  
Paul England

6/30/04  
Fecha

(Firma)  
Kenneth D. Ray

6/29/04  
Fecha

(Firma)  
Jamie Hunter

6/29/04  
Fecha

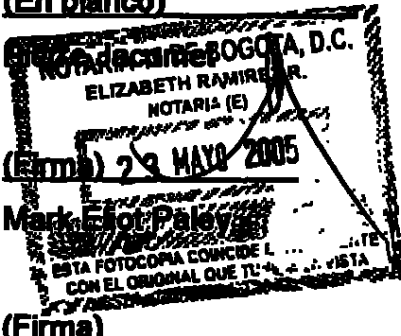
(Firma)  
Lonny Dean McMichael

MD

(En blanco)  
Fecha

(En blanco)  
Derek Norman LaSalle

(En blanco)  
Fecha

(En blanco)  
  
(Firma) Mark E. For. Paley

6/29/04  
Fecha

6/29/04

(Firma)

212

Fecha

Thekkthalackal Varugis Kurien

6/29/04

(Firma)

Fecha

David B. Cross

Registrado: 10/25/2004

**PATENTE**

**Carrete: 015284; Pantalla: 0856**

**CONTRAPORTADA: LOGO DEL DEPARTAMENTO DE COMERCIO DE  
ESTADOS UNIDOS – OFICINA DE PATENTES Y MARCAS REGISTRADAS**

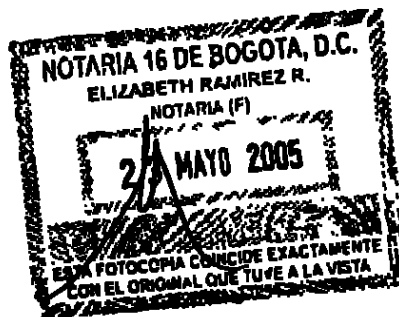
CERTIFICO QUE ESTA ES UNA TRADUCCION FIEL DEL INGLES AL  
ESPAÑOL DE UN DOCUMENTO QUE HE TENIDO A LA VISTA; QUE SOY  
TRADUCTORA E INTERPRETE OFICIAL ACREDITADA POR RESOLUCIONES  
0036 Y 526 DE LA UNIVERSIDAD NACIONAL DE COLOMBIA Y DEL  
MINISTERIO DE LA JUSTICIA RESPECTIVAMENTE, Y QUE ESTOY  
REGISTRADA ANTE EL MINISTERIO DE RELACIONES EXTERIORES DE  
COLOMBIA

Bogotá, mayo 16 de 2005

*Meryam Adam J.*

MERYAM ADAM J.  
C.C. 52.252.613

Traductora e Interprete Oficial  
Inglés-Español / Español-Inglés,  
Según Resolución 0526 de 11 de Junio de 2002  
Emitida por el Ministerio de Justicia y del Derecho  
Meryam Adam J.  
C.C. 52.252.613 Bnd





214

194

**PROTOCOLO**

**Expediente 05 - 51825**

**SUPERINDUSTRIA Y COMERCIO  
SISTEMA DE REGISTRO**

**MANTENIMIENTO DE PROTOCOLOS**

**Numero : 16069**

**Fecha : 04/08/1995**

**Conferido: MICROSOFT CORPORATION**

**País : US ESTADOS UNIDOS DE AMERICA**

**Ciudad : 234 REDMOND, WASHINGTON**

**Represent: S Sustituir: S Desistir: S**

**No. Radicacion: 95 34920 Clase: 0 Seev: 0 Seec: 0**

| Secuencia | Código | Ctr | Nombre                     |
|-----------|--------|-----|----------------------------|
| 1         | 77     | A   | CARDENAS NAVAS DARIO       |
| 2         | 3653   |     | CARDENAS CABALLERO EDUARDO |
| 3         | 5048   |     | CARDENAS MARTINEZ BERNARDO |

**Encontrados (1/1) registro(s)**

**Al contestar favor indique el número de  
radicación consignado en el sticker**

**Sede Centro: Carrera 13 No. 27-00 Pisos 2, 5, 7 y 10  
Computador: 3 820840 Fax: 3 505220  
Sede CAN: Tr. 40A No. 38-50 3153263 al 69  
Fax: 3 503220. Línea 9800-910 163  
Web: [www.ris.gov.co](http://www.ris.gov.co) e-mail: [info@ris.gov.co](mailto:info@ris.gov.co)  
Bogotá D.C. - Colombia**

Folio 195

2020  
Bogotá DC

Doctor(a)  
DARIO CARDENAS NAVAS  
Apoderado(a)

|                    |                |            |
|--------------------|----------------|------------|
| <b>REFERENCIA:</b> | Radicación No. | 05 - 51825 |
|                    | Trámite        | 002        |
|                    | Evento         | 001        |
|                    | Actuación      | 416        |
|                    | Oficio No.     | 433        |
|                    | Folios         | 001        |

Teniendo en cuenta que la solicitud de privilegio de patente que se tramita bajo el expediente indicado en la referencia, reúne los requisitos de forma establecidos en los artículos 26 y 27 de la Decisión 486 de la Comisión de la Comunidad Andina, y en las demás disposiciones vigentes sobre la materia, se envía el extracto de la solicitud a la Oficina de Comunicaciones para efecto de su publicación en la Gaceta de Propiedad Industrial.

De acuerdo con lo previsto por el artículo 40 de la Decisión 486/00, la publicación en este caso se hará a partir del día 27 de noviembre de 2006.

Cúmplase  
Dado en Bogotá DC, a los

09 JUN 2006

ALIX CARMENZA CÉSPEDES DE VERGEL  
Jefe de la División de Nuevas Creaciones

Al contestar favor indique el número de  
radicación consignado en el sticker

Sede Central: Carrera 13 No. 27-00 Pisos 2, 5, 7 y 10  
Commutador: 3 820840 Fax: 3 505220  
Sede CAN: Tr. 40A No. 38-50 3153265 al 69  
Fax: 3 505220. Línea 9800-910 165  
Web: [www.sic.gov.co](http://www.sic.gov.co) e-mail: [info@sic.gov.co](mailto:info@sic.gov.co)  
Bogotá D.C. - Colombia

**DIVISION DE NUEVAS CREACIONES**

**RADICACIÓN EXPEDIENTE No. 05-051825**

**EL EXTRACTO DE LA PRESENTE SOLICITUD DE PATENTE DE INVENCION**

**FUE PUBLICADO EN LA GACETA DE PROPIEDAD INDUSTRIAL No 570 DE**

**FECHA 30 DE NOVIEMBRE DEL 2006, EL TERMINO HABIL SEÑALADO POR**

**LA LEY EXPIRA EL 01 DE MARZO DEL 2007**

**NOTA: Dentro del plazo de los seis (6) meses siguientes a partir de la fecha de publicación en la gaceta de Propiedad Industrial, deberá solicitar y cancelar el examen de patentabilidad so pena de que la solicitud caiga en abandono, de acuerdo con lo establecido en el Artículo 44 de la Decisión 486/2000.**

**SE PRESENTARON OPOSICIONES POR PARTE DE TERCEROS.**

**SI    ☐    ☐**

**NO   ☐   ☒   ☐**

**Bogotá D.C., Junio de 2007**