Title: MECHANISM FOR ACCESS CONTROL OF COMPUTING SYSTEM IN PRE-OS STAGE

Abstract: A system that allows the owner or stake holder of the system to impose access control to the system, comprising a pre-operating system environment to perform a user check for the system; and a boot loader to load the pre-operating system environment based on authenticity of the pre-operating system environment and to boot the system in response to the success of the authorization check.
MECHANISM FOR ACCESS CONTROL OF COMPUTING SYSTEM IN PRE-OS STAGE

BACKGROUND

[0001] Currently, password checking may be used when booting a computer device. The mechanism may protect the Basic Input/Output System (BIOS) setting and usage of the computing device from unauthorized user. However, the authorization may only come from the owner of the computing device. If the computing device is bought through bank financing, the bank may not impose access control to the computing device. Further, it may be not easy to incorporate different authentication and/or authorization methods for the access control.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The invention described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0003] FIG. 1 illustrates an embodiment of a computing device.

[0004] FIG. 2 illustrates an embodiment of a system that may implement access control.

[0005] FIG. 3 illustrates an embodiment of a system that may be used in a pre-OS environment.
FIG. 4 illustrates an embodiment of a method or algorithm that may be used for access control.

DETAILED DESCRIPTION

The following description describes techniques to implement access control to a computing system in pre operating system (pre-OS) stage. The implementation of the techniques is not restricted in computing systems; may be used by any execution environments for similar purposes such as, for example, other digital/electronic equipment. In the following description, numerous specific details such as logic implementations, opcodes, means to specify operands, resource partitioning/sharing/duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the present invention. However, the invention may be practiced without such specific details. In other instances, control structures and full software instruction sequences have not been shown in detail in order not to obscure the invention.

References in the specification to "one embodiment", "an embodiment", "an example embodiment", etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.
009] Embodiments of the invention may be implemented in hardware, firmware, software, or any combination thereof. Embodiments of the invention may also be implemented as instructions stored on a machine-readable medium, which may be read and executed by one or more processors. A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), and others.

010] FIG. 1 shows an example embodiment of a computing device 100. The computing device 100 may comprise one or more processors 110 coupled to a chipset 120. The chipset 120 may comprise one or more integrated circuit packages or chips that couple the processor 110 to memory 130, Basic Input/Output System (BIOS) 140, storage device 150, and one or more I/O devices 160, such as, for example, mouse, keyboard, video controller, or other I/O devices of the computing device 100, etc.

011] Each processor 110 may be implemented as a single integrated circuit, multiple integrated circuits, or hardware with software routines (e.g., binary translation routines). The processor 110 may perform actions in response to executing instructions. For example, the processor 110 may executes programs, perform data manipulations and control tasks in the computing device 100, etc. The processor 110 may be any type of processor adapted to execute instructions from memory 130. For example, processor 110 may be a microprocessor, a digital signal processor, a microcontroller, or any other processors.
The memory 130 may comprise memory devices providing addressable storage locations that a memory controller 122 may read data from and/or write data to. The memory 130 may comprise one or more different types of memory devices such as, for example, dynamic random access memory (DRAM) devices, synchronous dynamic random access memory (SDRAM) devices, read-only memory (ROM) devices, or any other volatile or non-volatile memory devices. In one embodiment, the memory 130 may be arranged in a hierarchical manner. For example, the memory 130 may be arranged in channels, ranks, banks, pages, and columns. In another embodiment, the memory 130 may store one or more operating systems 134, such as, for example, Linux, Mac OS X, FreeBSD, Microsoft Windows or any other operating system.

The chipset 120 may receive transactions from the processors 110 and to issue transactions to the processors 110 via a processor bus 112. The chipset 120 may comprise a memory controller 122 that may read data from and/or write data to the memory 130 and/or issue transactions to the memory 130 via a memory bus 132. The chipset 120 may further comprise one or more I/O interfaces (not shown) to access one or more I/O devices 160 via buses 142 such as, for example, peripheral component interconnect (PCI) buses, accelerated graphics port (AGP) buses, universal serial bus (USB) buses, low pin count (LPC) buses, and/or other I/O buses. The I/O device 160 may include any I/O devices to perform I/O functions. Examples of the I/O device 160 may include controller for input devices (e.g., keyboard, mouse, trackball, pointing device), media card (e.g., audio, video, graphics), network card, and any other peripheral controllers.

In one embodiment, the storage device 150 may store archive information, such as code, programs, files, data, applications, or operating systems, or the like.
An example of the storage device 150 may comprise a tape, hard disk (HD) drive, a floppy diskette, a compact disk (CD) ROM, a flash memory device, any other mass storage device, any other magnetic storage media, any other optical storage media, any other non-volatile memory devices, or the like. The chipset 120 may further comprise one or more storage device interface 126 that may each access a storage device 150 via a bus 142.

In one embodiment, the BIOS 140 may comprise routines or drivers which the computing device 100 may execute. For example, the BIOS 140 may comprise routines or drivers to perform system initialization and/or configuration of the computing device 100. In another embodiment, the BIOS 140 may collect information that may be selectively used by an operating system. For example, the information may comprise a data structure that may be used by the operating system to look up one or more devices in the computing device 100. The BIOS 140 may further handle communications in the computing device 100, e.g., between software running on the computing device 100 and/or devices in the computing device 100, such as CPUs, disk drives, or printers, etc. In another embodiment, the BIOS 140 may comprise routines or drivers which the computing device 100 may execute to communicate with one or more components of the computing device 100. In another embodiment, the BIOS 140 may be implemented with a firmware.

For example, the BIOS 140 may comprise a legacy BIOS, extensible firmware interface (EFI) BIOS, or any other BIOS. The chipset 130 may comprise a BIOS interface 124 that may access the BIOS 140 via a bus 142.

In another embodiment, the BIOS 140 may be implemented as a single integrated circuit, multiple integrated circuits, hardware with software routines, or one or more executable modules with BIOS codes. In one embodiment, the BIOS...
140 may access a secure storage device 170. The secure storage device 170 may store secure data associated with the BIOS 140, e.g., a hash value, etc. An example of the secure storage device 170 may comprise a flash memory device, or any other non-volatile memory devices. For example, the secure storage device may be implemented in a trusted platform module (TPM). While FIG. 1 shows that the computing device 100 comprises the secure storage device 170, in another embodiment, the secure storage device 170 may be absent in the computing device 100 and the computing device 100 may comprise a BIOS memory to store BIOS code and data, and/or any other secure data associated with the BIOS 140.

The BIOS memory may be implemented with non-volatile memory devices, such as read-only memory (ROM) devices, flash memory, or any other memory devices. The BIOS 140 may further contain a BIOS USB driver and other drivers. In yet another embodiment, the BIOS 140 itself may store BIOS code or data, and/or any other secure data associated with the BIOS 140.

The computing device 100 may communicate with a remote device 190 via a network 180. The chipset 130 may comprise a network controller 128 to control the communication between the computing device 100 and the network 180 via, for example, a network card, etc. In some embodiments, the computing device 100 may communicate with a different number of remote devices via a different number of networks. The computing device 100 may link to the network 180 via a network cable, wirelessly, or in other ways. The network 180 may comprise a cable network, a wired network, a wireless network, a local area network (LAN), a wide area network (WAN), a Bluetooth network, Internet, World Wide Web, or any other networks.
While FIG. 1 shows one example of the computing device 100, other embodiments may employ one or more other devices in the computing device 100.

FIG. 2 illustrates an example embodiment of a system 200 that may impose access control to a computing system, e.g., the computing device 100 as shown in FIG. 1. In one embodiment, the system 200 may comprise a boot loader 210, pre-OS environment or module 220 and a secure storage device 230. The boot loader 210 may boot to any bootable device that may be designated by a user in BIOS configuration. In one embodiment, the boot loader 210 may comprise routines or codes that the computing device 100 may execute to boot the bootable devices. In another embodiment, the boot loader 210 may be implemented with a firmware. In yet another embodiment, the boot loader 210 may be implemented with an executable module that comprises codes or routines executable by the computing device 100, such as a chip, an integrated circuit, or any other executable module. Examples of the bootable devices may comprise one or more storage devices, e.g., storage device 150, such as hard drive and/or bootable partition of the hard drive, CD ROM, or the like. In another embodiment, the boot loader 210 may load one or more operating systems on the computing device 100. Although FIG. 2 shows that the boot loader 210 is located individually, in some embodiments, the boot loader 210 may be located or stored in the BIOS 140. For example, the BIOS 140 may initiate the boot loader 210 after initializing the computing device 100.

The pre-OS management environment 220 may be stored in a partition of hard disc drive of the computing device 100. In another embodiment, the pre-OS management environment 220 may be stored in any other storage devices 150, e.g., CD ROM, etc. In yet another embodiment, the pre-OS management
environment 220 may be implemented as a part of the BIOS 140. In still another embodiment, the pre-OS management environment 220 may be implemented with a firmware. In another embodiment, the pre-OS management environment 220 may be implemented with an executable module, such as a chip or an integrated circuit, that comprises codes or routines executable by the computing device 100.

The hard disc drive partition that contains the pre-OS management environment 220 may comprise a code region 222 and a data region 224. In one embodiment, the code region 222 may store code, data, program or any other information or software to perform authentication and/or authorization on a user of the computing device 100. In another embodiment, the code region 222 may store data that may be static or may not be modified. The data region 224 may store data or any other information that may be used to check the user of the computing device 100 and the right of the user to use the computing device 100. In one embodiment, the data region 224 may store data that may be dynamic or may be modified. The pre-OS management environment 220 may have full access to resources of the computing device 100, such as network connection, hard disk drive, one or more other storage devices, and one or more USB devices. The pre-OS management environment 220 may implement resource demanding, authentication and/or authorization mechanisms based on one or more algorithms and/or codes. The data region 224 may further store other data to support the pre-OS management environment 220. For example, the data region 224 may store network configuration data that may be used to connect the pre-OS management environment 220 to a remote server.

The secure storage device 230 may be optional in the system 200. The secure storage device 230 may store secure data associated with the BIOS 140 and/or
any other secure data associated with the computing device 100. The secure storage device 230 may be accessible by the boot loader 210. In another embodiment, if the secure storage device 230 is absent, the secure data may be stored in the BIOS 140 or in a BIOS memory and may be accessible by the boot loader 210.

Although FIG. 2 shows that the boot loader 210 and the pre-OS management environment 220 are located individually, in some embodiments, the boot loader 210 and the pre-OS management environment 220 may be implemented on the same executable module. In another embodiment, the boot loader 210 and the pre-OS management environment 220 may be implemented as a part of the BIOS 140.

FIG. 4 illustrates an example embodiment of a method that may be used for access control to the computing device 100. In one embodiment, the method of FIG. 4 may be used in the system 200 as shown in FIG. 2. However, some embodiments may use any execution environments for similar purposes. Referring to FIG. 4, in block 402, the boot loader 210 may start to boot the computing device 100, e.g., in response to initialization of the computing device 100. In block 404, the boot loader 210 may search a hard disc drive of the computing device for a hard disc drive partition that may contain the pre-OS management environment 220. In another embodiment, the boot loader 210 may scan one or more storage devices of the computing device 100 for the pre-OS management environment 220.

The boot loader 210 may shut down the computing device 100 (block 414), in response to determining that the partition does not exist or the pre-OS management environment 220 is absent. On the contrary, in response to
determining that the hard disc drive comprises the partition that contains the
pre-OS management environment 220 or the pre-OS management environment is
available, the boot loader 210 may determine the authenticity or safety of the
pre-OS management environment 220 (block 406). For example, the boot loader
210 may execute code, data, or any other program to authenticate the pre-OS
management environment 220. For another example, the code or data may be
stored in the BIOS 140. In one embodiment, the boot loader 210 may scan the
code region 222 of the partition to obtain first secure data stored in the code region
222. The boot loader 210 may further fetch second secure data stored in BIOS 140
or a BIOS memory and compare the first secure data with the second secure data.
For another example, the boot loader 210 may fetch the second secure data that is
stored in the secure storage device 230 and compare the first secure data with the
second secure data, if the secure storage device 230 is present. In another
embodiment, the boot loader 210 may perform a hash value check. For example,
the boot loader 210 may scan the code region 222 of the partition and compute a
hash value from data stored in the code region 222 by using, e.g., SHA-1 or MD5
hash algorithm. The boot loader 210 may further compare the calculated hash
value with a known hash value that may be fetched from the BIOS 140, the BIOS
memory, or the secure storage device 230.

In block 408, the boot loader 210 may load the pre-OS management
environment 220, in response to determining that the pre-OS management
environment 220 is authentic. For example, the boot loader 210 may load pre-OS
management environment 220 to the memory 130 of the computing device 100
from the hard disc drive or any other storage device, or from the BIOS 140. The
boot loader 210 may transfer process to the pre-OS management environment 220.
In another embodiment, the boot loader 210 may pass other secure data that is stored in BIOS 140, the BIOS memory, or the secure storage device 230 to the pre-OS management environment, e.g., the data region 224. In one embodiment, the boot loader 210 may determine that the authenticating of the pre-OS management environment 220 is successful, in response to determining that the first secure data matches the second secure data. In another embodiment, the boot loader 210 may determine that the pre-OS management environment 220 is authentic, in response to determining that the calculated hash value is equal to or matches the known hash value stored in the secure storage device 230 or the BIOS 140. On the contrary, if the secure data or hash value comparison fails, it may indicate that, e.g., the code region 222 has been changed or pre-OS management environment 220 has been compromised or not authentic. The boot loader 210 may shut down or halt the computing device 100 (block 414).

In block 410, the loaded pre-OS management environment 220 may perform a check on a user of the computing device 100. The check may comprise an authentication check and/or an authorization or user right check on the user. For example, the pre-OS management environment 220 may execute an authentication program or codes to determine whether the user is authentic, e.g., who the user is. The pre-OS management environment 220 may execute an authorization program or codes to check user right, e.g., whether the user is authorized to use the computing device 100, and/or how long the user is authorized to use the computing device, and/or how the user is authorized to use the computing device, etc. In one embodiment, the pre-OS management environment 220 may perform the authentication check and/or the authorization check based on checking credential, such as digital document, digital signature,
USB key, or any other data, from the data region 224 and/or one or more external sources.

In one embodiment, the pre-OS management environment 220 may check if the user has a proper or valid digital certificate. The digital certificate may be stored in a secure storage space of the pre-OS management environment 220, such as the data region 224. For another example, the digital certificate may be stored in the code region 222. In another embodiment, the digital certificate may be provided by a remote control server through a secured network connection that couples to a third party's system to impose access control to the computing device. In response to determining that the digital certificate is trusted, the pre-OS management environment 220 may further determine authorized user right based on the trusted digital certificate, e.g., whether the user is authorized to use the computing device, and/or how much rights the user has, etc.

For example, an example of the third party may include a bank that sold the computing device 100 through bank financing, such as through paying loan installment, a lender that rented the computing device, a service provider that provided service for the computing device, or an OEM that maintains the computing device. In response to the digital certificate being expired or corrupted, the pre-OS management environment 220 may request a new digital certificate from the remote control server through the secured network connection based on payment status of the computing device. In one embodiment, the third party's system may monitor the payment status and/or issue a new digital certificate in response to the user paying a corresponding fee, such as the loan installment or any other fee, to the third party. In response to the new digital certificate is available, the user may fetch the new digital certificate from the remote control
server for the authentication and/or authorization check by the pre-OS management environment 220. On the contrary, in response to the user failing to pay the loan installment, the rental fee, the service fee, or the maintenance fee to the third party, the new digital certificate may be unavailable.

In another embodiment, the pre-OS management environment 220 may check whether a smart card that the user inserts into a smart card reader is valid. For example, the pre-OS management environment 220 may check whether the smart card is expired or the smart card is issued by the associated third party. In response to determining that the smart card is valid, the pre-OS management environment 220 may perform an authorization or user right check. For example, the pre-OS management environment 220 may determine whether the user is authorized to use the computing device based on information stored in the smart card. The user may get the smart card that is authorized by the third party based on the payment status of the corresponding fee. For another example, the user may buy a fee charging card that may be distributed by the third party for the user to pay the corresponding fee to the third party offline. The pre-OS management environment 220 may further determine the authorized user right, in response to the payment of the corresponding fee.

In yet another embodiment, the pre-OS management environment 220 may prompt the user to type in data or code(s), e.g., a password, a security key or any other data. The pre-OS management environment 220 may determine authenticity of the data or code from the user. The pre-OS management environment 220 may further perform an authorization check based on the data or code. In one embodiment, the user may get the code or data from the third party. In another
embodiment, the user may get the code or code from a remote server via a network connection or offline.

In one embodiment, a remote server may perform the authentication and/or authorization check through a network. For another example, authentication and/or authorization codes or programs may be executed offline through an algorithm, such as a pre-installed secretary algorithm. In another embodiment, one of the authentication check and the authorization check may be omitted; however, in some embodiments, the two checks may be combined.

The pre-OS management environment 220 may shut down the computing device (block 414), in response to determining that the authentication check and/or the authorization check on the user is failed. On the contrary, in response to determining that the authentication check and/or the authorization check is successful, e.g., the user has right to use the computing device 100, the pre-OS management environment 220 may hand the control back to the boot loader 210 (block 412). The boot loader 210 may continue the normal boot process. For example, the boot loader 210 may continue to boot any bootable device that may be designated by the user in the configuration of the BIOS 140. In another embodiment, the boot loader 210 may store a new hash value and/or other new secure data that are produced in the boot process in the secure storage device 230 or the BIOS 140 or the BIOS memory. For another example, the boot loader 210 may transfer process to an operating system loader in a device that is booted by the boot loader 210 to load an operating system of the computing device 100. In another embodiment, the boot loader 210 may load an operating system that is stored in the computing device 100. For example, the boot loader 210 may boot one or more storage devices 150 that may each comprise an OS loader to load
one or more operating systems. For another example, the boot loader 210 may load one or more operating systems directly. In one embodiment, the operating system may provide a runtime environment to support computer applications that may be used by user. In another embodiment, the operating system may enable use of the computing device.

While the method of FIG. 4 is illustrated as a sequence of operations, in some embodiments, the illustrated operations may be performed in a different order. While the method is described with reference to a computing device, in some embodiments, the method may be used for any other user devices for similar purposes.

FIG. 3 illustrates an example embodiment of a system 300 that may be used in the pre-OS management environment 220. The system may interact with the boot loader 210. In one embodiment, the system 300 may comprise a credential and/or authorization check framework that may comprise authentication and/or authorization data or codes to implement different methods, algorithms and/or policies for access control of a computing device. For example, the system 300 may comprise one or more modules 310-1 through 310-N, such as checking codelets, that may perform authentication and/or authorization for accessing the computing device. The methods, algorithms and/or policies for access control of the computing device may be implemented in the one or more modules 310-1 through 310-N. If a new algorithm or policy for access control is desired, an implementer may only write one or more of the modules 310-1 through 310-N.

The system 300 may further provide one or more modules that may be used by the access control codes or data to use one or more system resources, such as network, USB peripherals, and secure data storage. Example of the modules may
comprise network access module 322, security/public key infrastructure (PKI) Application Programming Interface (API) 324, secure storage API 326, device I/O API 328, or any other modules. These modules may be exposed to the checking codelets 310-1 through 310-N to facilitate access to the system resources.

While certain features of the invention have been described with reference to embodiments, the description is not intended to be construed in a limiting sense. Various modifications of the embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.
What is claimed is:

1. A system comprising:
   
a pre-operating system environment to perform a user check for the system;

and

5  
a boot loader to load the pre-operating system environment based on authenticity of the pre-operating system environment and to boot the system in response to success of the user check.

2. The system of claim 1, further comprising:

10  
a secure storage device that is accessible to the boot loader to store first secure data associated with the system that is used to determine the authenticity of the pre-operating system environment.

3. The system of claim 1, further comprising:

15  
a memory that couples to the boot loader to store a known hash value associated with the system that is used to check the authenticity of the pre-operating system environment.

4. The system of claim 1, further comprising:

20  
one or more bootable devices that are bootable by the boot loader to load one or more operating system associated with the system.

5. The system of claim 1, the boot loader further to shut down the system, in response to determining that the pre-operating system environment is absent.
6. The system of claim 1, the boot loader further to
check the authenticity of the pre-operating system environment based on a
comparison between first secure data associated with the system and second
secure data obtained from the pre-operating system environment.

7. The system of claim 6, the boot loader further to
halt the system, in response to determining that the comparison fails.

8. The system of claim 1, the boot loader further to
compare a known hash value associated with the system and a second
hash value calculated from data that is stored in a code region of the pre-operating
system environment to determine the authenticity of the pre-operating system
environment.

9. The system of claim 8, the boot loader further to
shut down the system, in response to determining that the known hash
value mismatches the second hash value.

10. The system of claim 1, the boot loader further to
load the pre-operating system environment to check whether a user is
authorized to access the system based on a credential associated with the user, in
response to determining that the pre-operating system environment is authentic.

11. The system of claim 1, the pre-operating system environment further to
halt the system, in response to determining that the user check fails.

12. The system of claim 1, further comprising:

an operating system that enables use of the system, in response to the boot loader booting a bootable device of the system to load the operating system.

13. A method, comprising:

- determining authenticity of a pre-operating system environment of a user device based on a check on first secure data of the pre-operating system environment; and

- loading the pre-operating system environment to determine whether a user is authorized to use the user device, in response to determining that the pre-operating system environment is authentic.

14. The method of claim 13, further comprising:

halting the user device, in response to determining that the pre-operating system environment is absent in the user device.

15. The method of claim 13, further comprising:

comparing the first secure data of the pre-operating system environment and second secure data associated with the user device; and

shutting down the user device, in response to determining that the first secure data mismatches the second secure data.

16. The method of claim 13, further comprising:
checking whether a known hash value associated with the user device matches a second hash value obtained from the first secure data of the pre-operating system environment; and

halting the user device, in response to determining that the known hash value is unequal to the second hash value.

17. The method of claim 13, further comprising:
determining whether the user is authorized to use the user device based on a payment status of the user device.

18. The method of claim 13, further comprising:
determining whether the user is authorized to use the user device based on a credential provided by a bank system.

19. The method of claim 13, further comprising:
transferring control to the pre-operating system environment to determine whether the user has an authorized certificate to use the user device, in response to determining that the pre-operating system environment is loaded successfully.

20. The method of claim 13, further comprising:
checking a code from the user to determine whether the user is authorized to use the user device.

21. The method of claim 13, further comprising:
loading an operating system to enable use of the user device, in response

to the authorization determination on the user being successful.

22. A machine readable medium comprising a plurality of instructions that in
response to being executed result in a computing device
determining whether the computing device contains an authentic
pre-operating system environment based on a hash value check; and
loading the pre-operating system environment to control access to the
computing device based on a user check, in response to determining that the hash
value check is successful.

23. The machine readable medium of claim 22, further comprising a plurality
of instructions that in response to being executed result in the computing device
halting the computing device, in response to determining that the
pre-operating system environment is absent in the computing device.

24. The machine readable medium of claim 22, further comprising a plurality
of instructions that in response to being executed result in the computing device
comparing a first hash value stored in the computing device and a second
hash value calculated from data that is stored in a secure region of the
pre-operating system environment.

25. The machine readable medium of claim 22, further comprising a plurality
of instructions that in response to being executed result in the computing device
comparing a digital certificate from a user of the computing device with data
stored in a data region of the pre-operating system environment to perform the
user check.

26. The machine readable medium of claim 25, further comprising a plurality
of instructions that in response to being executed result in the computing device
shutting down the computing device, in response to determining that the
digital certificate is expired and a new digital certificate is unavailable.

27. The machine readable medium of claim 25, further comprising a plurality
of instructions that in response to being executed result in the computing device
booting the computing device, in response to determining that the digital
certificate from the user is valid.

28. The machine readable medium of claim 22, further comprising a plurality
of instructions that in response to being executed result in the computing device
checking whether a user of the computing device has an authorized code to
use the computing device under the pre-operating system environment.

29. The machine readable medium of claim 28, further comprising a plurality
of instructions that in response to being executed result in the computing device
booting one or more bootable devices of the computing device, in response
to determining that the user has the authorized code.
30. The machine readable medium of claim 28, further comprising a plurality of instructions that in response to being executed result in the computing device halting the computing device, in response to determining that the authorized code is unavailable.

31. A system, comprising:
   a processor;
   a pre-operating system environment that couples with the processor to perform access control to the system based on an check on a user of the system;
   a boot loader to boot the system, in response to the success of the check on the user; and
   an operating system to support the system, in response to the boot loader booting the system.

32. The system of claim 31, the boot loader further to:
   compare first secure data associated with the system with second secure data associated with the pre-operating system environment to determine authenticity of the pre-operating system environment; and
   transfer control to the pre-operating system environment, in response to determining that the first secure data matches the second secure data.

33. The system of claim 31, further comprising:
   a secure storage device that is accessible by the boot loader to store a known hash value associated with the system,
the boot loader further to compare the known hash value with a second
hash value computed from data stored in a code region of the pre-operating
system environment to determine whether the pre-operating system environment
is authentic.

34. The system of claim 31, the boot loader further to
halting the use of the system, in response to determining that the
pre-operating system environment has been compromised.

35. The system of claim 31, the pre-operating system environment further to
shutting down the system, in response to determining that the user is
unauthorized to use the system.

36. The system of claim 31, further comprising
an operating system loader to load the operating system, in response to the
boot loader booting a device of the system that comprises a loader to load the
operating system.
FIG. 3

Credential & Authorization Checking Framework

- Checking Codelet 310-1
- Checking Codelet 310-2
- Checking Codelet 310-3

- Network Access 322
- Secure Storage API 326
- Device I/O Port 328

- Security/PKI API 324
- Other Drivers 338
- Network Stack 336
- File System 334
- USB Driver 332
Begin

Starting to Boot

Pre-OS Environment Exists?

Pre-OS Environment is Authentic?

Loading Pre-OS Environment

User is Authorized?

Continuing to Boot

Shutting Down

End

FIG. 4
# INTERNATIONAL SEARCH REPORT

**A. CLASSIFICATION OF SUBJECT MATTER**

G06F 9/44S (2006.01) i  
According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

G06F 1/00 (2006.01) i, H04L 9/32 (2006.01) i, G06F 3/00(2006.01)i

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

CNPAT

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, WPI, EPODOC, PAJ: access, control, pre-OS, user, authorize, boot, load, computer, device, system, BIOS, authenticity, confirmation, verification, use;

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
</table>

☐ Further documents are listed in the continuation of Box C.  ☑ See patent family annex.

* Special categories of cited documents:

  "A" document defining the general state of the art which is not considered to be of particular relevance
  "E" earlier application or patent but published on or after the international filing date
  "I" document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)
  "O" document referring to an oral disclosure, use, exhibition or other means
  "P" document published prior to the international filing date but later than the priority date claimed

  "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principal or theory underlying the invention
  "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
  "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
  "&" document member of the same patent family

Date of the actual completion of the international search  

Date of mailing of the international search report  
28 · DEC 2006 (28 · 12 · 2006)

Name and mailing address of the ISA/CN  
The State Intellectual Property Office, the P.R.China  
6 Xitucheng Rd., Jiemian Bridge, Haidian District, Beijing, China 100088  
Facsimile No. 86-10-62010451

Authorized officer  
[Signature]

Telephone No. 86-010 62084953

Form PCT/ISA/210 (second sheet) (April 2005)
<table>
<thead>
<tr>
<th>Patent Documents referred in the Report</th>
<th>Publication Date</th>
<th>Patent Family</th>
<th>Publication Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN,A,1549087</td>
<td>24.11.2004</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>WO,A1,2005091106</td>
<td>29.09.2005</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>