(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 February 2008 (07.02.2008)

PCT

(10) International Publication Number
WO 2008/015412 A1

(51) International Patent Classification:
G06F 21/02 (2006.01)

(21) International Application Number:
PCT/GB2007/002900

(22) International Filing Date: 31 July 2007 (31.07.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PI 20063707 31 July 2006 (31.07.2006) MY
0621338.3 26 October 2006 (26.10.2006) GB

(71) Applicant *(for all designated States except US)*: **BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY** [GB/GB]; 81 Newgate Street, London Greater London EC1A 7AJ (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CHOONG, Khong, Neng** [MY/MY]; 29 Taman K.s., Negeri Sembilan, Mantin, 71700 (MY). **LOW, Andy, Lock, Yen** [MY/MY]; 242-G, Taman Sinn, Semabok, Melaka, 75050 (MY).

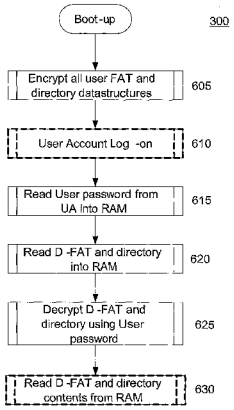
(74) Agent: **WILLIAMSON, Simeon, Paul**; BT Group Legal Intellectual Property Department, Ppc5a, Bt Centre, 81 Newgate Street, London Greater, London EC1A 7AJ (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: SECURE DATA STORAGE



(57) Abstract: The present invention relates to secure data storage on hard disk drives or other portable or fixed media for computer systems and computing devices. The present invention provides a method of operating a computing device (100) having working memory (120) and coupled to a storage medium (130), the method comprising: instantiating an encrypted file allocation datastructure (560, 563) stored on the storage medium into the working memory ((620), the file allocation datastructure comprising storage locations on the storage medium for a number of files; decrypting the file allocation datastructure in working memory (625); reading the decrypted file allocation datastructure (570, 573) in order to locate a said file within the storage media (630); encrypting the file allocation datastructure in working memory upon updating and saving the file; writing the encrypted file allocation datastructure back to the HDD.

WO 2008/015412 A1

SECURE DATA STORAGE

Field of the Invention

The present invention relates to secure data storage on hard disk drives or other portable or fixed media for computer systems and computing devices.

Background of the Invention

Portable, home or work computer systems store numerous data, from personal records, multimedia contents, to company confidential information. Given current media storage (eg hard disk drive) sizes, a large amount of such data can be stored on a single computing device. Furthermore many storage media are portable and can be transferred between multiple computing devices. Therefore keeping the data intact, safe and secure is an important challenge.

Operating systems such as Microsoft Windows™ can be configured to securely partition a hard disk drive (HDD) for different users, so that for example a username and password is required for access to a protected partition. Examples include User Account management in Windows XP and GNU Linux. However these security systems can be by passed by booting-up a computing device using an operating system from a different HDD or a bootable CD-ROM, and then inspecting the partitioned HDD without the operating system enforced partitioning.

US2003/0204754 describes concealment of a portion of the storage media (eg HDD) above a certain boundary pointer. A bios level password entry algorithm is used to set the boundary pointer depending on the user. For example a system administrator may have full access to the operating system, whereas a normal user may be restricted from this area of the storage media by lowering the boundary pointer.

WO04/086228 describes a security device which intercepts read/write instructions to the HDD, and allows or disallows them depending on whether the current user has access to the instruction destination on the HDD.

Summary of the Invention

In one aspect, the present invention provides a multi-user password protection arrangement for a user partitioned storage medium (eg HDD), and which is enabled at the firmware (eg BIOS) level. The system or method assigns each user a designated user partition dynamically upon start-up of a device (eg PC) using the storage medium.

There is provided a method and system of securely storing data on a storage medium such as a hard disk drive which stores data for a plurality of users and which is coupled to a computing device. Firmware or BIOS level program instructions are used following powering on of the computing device to modify the storage medium partition information passed to the operating system or other software program instructions to be loaded from the storage medium depending on which user is authenticated by a login or other authentication procedure. The software program instructions can then be loaded and take over execution from the firmware program instructions, and will be passed storage medium partition information dependent on the authenticated user. This secures user data stored in respective partitions from being accessed by a non-authenticated user, as any other user partitions are hidden from the operating system.

Storage medium partition information is stored in a storage medium partition datastructure such as a partition table which is loaded upon computing device boot-up. Subsequent programming modules such as the operating system require this information for finding files on the storage medium. Thus when parts of the storage medium corresponding to non-authenticated users are not included in this partition datastructure, they are hidden to the operating system.

The term partition is used in its standard computing sense to refer to a division of a storage medium such as a hard disk drive into self-contained logical units that can be interrogating independently of each other by a program such as an operating system executing on a computing device coupled to the storage medium.

In an embodiment, a storage medium partition datastructure such as a partition table (PT) stored on the storage medium is loaded into memory and then modified prior to passing the boot-up execution control to an operating system. The partition table in the working memory is modified to incorporate a record or entry relating to the current or authenticated user. For example a partition table may have start and end addresses corresponding to the authenticated user's partition added or written. Subsequent software program instructions such as an operating system (eg Windows™) from an HDD which are to be loaded and executed on the computing device require information about the memory layout of the storage medium containing the user data. This is transferred to the software program instructions (OS) using the storage medium partition datastructure (eg PT). By incorporating a PT entry only for an authenticated user, and not having PT entries for other users, whatever operating system is booted up, it will only see the partitions indicated in the modified partition table – ie only the authenticated user. Any other partitions associated with different users will be invisible to the operating system.

In an embodiment a user partition datastructure or user data table (UDT) is used to store start and end addresses (eg cylinder-head-sector of HDD) of each user's partition. This may be encrypted (and/or hidden to the OS) when stored on the storage medium. The storage medium partition datastructure such as a partition table (PT) is modified by writing in an additional partition entry with the user partition's start and end addresses. The storage medium partition datastructure (PT) is typically loaded into working memory (eg RAM) prior to modification such that the PT stored on the HDD retains no or a single non-user specific main partition entry; for example a C:/ drive containing the OS or other software program instruction code. When the operating system starts, it reads the PT to determine the useable areas of the HDD, and therefore is unable to see any non-authenticated user's partitions.

In an embodiment, a computing device such as a personal computer (PC), personal digital assistant (PDA), or Smart-phone executes firmware program instructions such as BIOS code following power-on of the device. These instructions are typically stored in a non-volatile memory module such as a ROM store which is securely integrated in the computer device, for example being an integrated circuit soldered onto the mother board of the device. The firmware instructions run a user log-in procedure typically prompting a user for a username and password. This is matched against username and password entries for a number of predetermined users, typically stored in the user partition datastructure (UDT). A successful match authenticates one of the predetermined users, and results in the partition start and end addresses for the authenticated user being retrieved from the UDT. These are read into the working memory, typically RAM, and used to modify the PT which is also loaded into the working memory.

The firmware will typically then hand over control of the computing device to a boot loader program which loads the operating system (OS) from the storage medium (or possibly an inserted bootable CD-ROM) into the working memory and executes this OS. The OS requires the PT in the working memory as the basis to determine the structure of the storage medium (HDD). Thus the OS relies upon the modified PT for its "understanding" of what is on the HDD, and any storage medium areas not indicated by the partition table or equivalent storage medium partition datastructure will be invisible to the OS and any user interacting with this.

In an embodiment, each partition or part of a partition such as the file allocation table (FAT) or equivalent file allocation datastructure is encrypted, and the system decrypts the user's partition (or partition part) in response to correct entry of the user's password. The decrypted datastructure will then typically over-write the encrypted datastructure stored on the HDD, as the operating system normally accesses this datastructure directly from the HDD rather than from working memory (eg RAM). The embodiment may initially be arranged to encrypt all file allocation datastructures, other specified parts of each user partition, or the partitions themselves prior to user log-in. This ensures that none of these

non-authenticated user partitions are available following boot-up of the OS. Alternatively a BIOS level power-down routine may be used to provide this functionality.

In another aspect, the present invention provides a method and system for providing secure user data on a storage medium having one or more user partitions and which is coupled to a computing device. Each partition of a storage medium includes a file allocation datastructure which comprises the locations on the storage medium of respective files or other datastructures. The file allocation datastructure or equivalent file is encrypted on the storage medium, instantiated into working memory on the computing device and decrypted. Access to the file allocation datastructure by any program instructions executing on the computing device is then made using the decrypted file allocation datastructure in working memory instead of in the storage medium as is usual. Because the files stored on the storage medium cannot be accessed without the respective file allocation datastructure, a user authentication procedure is used to protect the user data in these files by decrypting the file allocation datastructure only in response to successful authentication of a respective user. This arrangement may be carried out fully at the operating system level, for example following boot-up, or it may be combined with the first aspect in order to provide two levels of user authentication –BIOS and OS levels.

In an embodiment the system or method decrypts one or more file allocation datastructures (eg FAT) in response to correct entry of a password. This approach may be applied in the case of a single user partition (or un-partitioned storage medium), the user being required to successfully log-in during the firmware/BIOS stage following start-up of the computing device, and/or the OS boot-up stage, before the file allocation datastructure is decrypted – for example using a key stored on the firmware memory module or a user account management file.

There is also provided a storage medium for use with a computing device, the storage medium comprising an encrypted file allocation datastructure comprising storage locations on the storage medium for a number of files.

Brief Description of the Drawings

Embodiments will now be described, by way of example only and without intending to be limiting, in which:

Figure 1 is a schematic of the memory architecture of a computing device according to an embodiment;

Figure 2 is a flow diagram of a method of allocating a partition to a user according to an embodiment;

Figure 3 is a flow chart of an authentication method according to an embodiment;

Figure 4 is a schematic illustrating in more detail a user data table according to an embodiment;

Figure 5 is a schematic illustrating memory architecture for a computing device according to an embodiment;

Figure 6 is a flow diagram illustrating a method of booting up the computing device of figure 5;

Figure 7 is a flow diagram illustrating a method of reading a file allocation datastructure in the computing device of figure 5;

Figure 8 is a flow diagram illustrating a method of writing a file allocation datastructure of the computing device of figure 5; and

Figure 9 is a flow diagram illustrating a method of booting up the computing device of figure 5 according to another embodiment.

Detailed Description

Figure 1 illustrates the memory architecture at a high level for a computing device such as a personal computer (PC). The computing device 100 comprises firmware 110 such as basic input output software (BIOS) which is typically stored on read only memory (ROM) securely integrated with the device 100. The device also comprises random access memory (RAM) 120 which is typically volatile and used as the processor's working memory into which various software program instructions such as the operating system 123 and data such as a partition table 125 are loaded or instantiated and executed or modified. The device 100 also either comprises or is coupled to a non-volatile storage medium or memory 130 such as a hard disk drive (HDD) which may be partitioned into a number of partitions 141, 147 - in this embodiment an operating system partition (C:/) 141 and a number of user partitions (D1:/, D2:/, D3:/) 147. Each partition is self-contained as far as the operating system is concerned, and comprises its own file allocation datastructure 143, such as a file allocation table (FAT) for pre-XP Window's based systems, or a Master File Table (MFT) for an XP based system. Each partition also comprises its own boot sector code 139, root directory 148, and data area 149 for storing data and programs.

The operating system partition (C:/) 141 comprises the operating system program code (OS) 133 which can be loaded onto the RAM 120 and executed (123), as well as a master boot record (MBR) 137 in a Window's based system. Equivalent structures exist in non-Window's based systems. The MBR 137 is the first memory location on the HDD 130 at which the system or execution thread looks after the end of the BIOS 110 start-up routines and contains a partition table 135, a master boot code 131 which locates, loads, and executes the active partition's boot sector code 139, as well as an initial jump instruction 132 which loads and executes the main boot code 131.

The BIOS or firmware instructions within the firmware module 110 load the partition table 135 from a known standard location on the HDD 130 into the working memory 120

as well as performing various power on self test (POST) routines. The BIOS then passes control to the MBR 137. The execution thread moves on to the MBR jump instruction 132, the master boot code 131, then the boot sector code (139) of the active partition (C:/) which comprises boot loader code for loading software program instructions (the OS) 133 stored on the HDD 130 into the RAM (instantiated as 123) for execution. The partition table 125 loaded into RAM is read by the OS 123 to determine the useable areas of the HDD 130. Typically the partition table will include the start and end addresses (head-cylinder-sector) within the HDD 130 of each partition, including the main or OS partition (C:/) 141 and another user partition (D:/) 147.

The embodiment additionally comprises a user data table (UDT) 145 or other user partition datastructure which contains the start and end addresses (head-sector-cylinder) for a number of user partitions (D1:/ - D3:/) 147. Depending on the user of the device 100, the start and end addresses from the UDT 145 are used to modify the partition table 125 to include the specific user partition entry (eg D2:/) for the corresponding user. Thus when the operating system starts up, it will only see the operating system partition (C:/) 141 and one of the user's partitions (D2:/) 147, and will not see the other user partitions (D1:/, D3:/) 147. These will effectively be hidden from the operating system and hence the user, so the user will not be able to access the user partitions of other users and their contents will remain secure. This is the case even if a user tries to boot-up an operating system from a different storage medium or CD-ROM, as the modification of the partition table (PT) or equivalent storage medium partition datastructure 125 always occurs before loading and/or execution of the OS which requires the PT 125 in order to function properly, and obtain some of the data it requires for proper start up and operation.

The UDT 145 may be stored on the HDD 130 and will typically be encrypted in order to prevent unauthorised access. Alternatively the UDT 145 may be securely stored unencrypted on the device 100, for example within the firmware 110. Typically in a system with a portable HDD, the UDT 145 will be kept within the HDD so that it can be interrogated irrespective of which device the HDD is inserted into.

Typically each user partition (D1:/ - D3:/) 147 or a part of this partition is encrypted in order to prevent access to these partitions 147 when the storage medium 130 is coupled to a different computing device 100 without the BIOS level storage medium partition datastructure (eg PT) modification. The relevant partition can then be decrypted as part of the BIOS level PT modification routine. Therefore decryption will not occur if this BIOS user login routine is not carried out. This option is useful where the storage medium 130 is portable and can be moved to different computing devices. The part of the partition that is encrypted may be a file allocation datastructure for each partition such as the file allocation table (FAT) or master file table (MFT), and which is only decrypted by the firmware upon successful entry of a user password. The encryption passwords may be stored in the UDT 145 and may be different for each user partition 147, or a single encryption password may be used and stored within the firmware 110 for example.

The thread of execution starts in the BIOS 110, modifying the partition table 125 in the working memory 120, before completing the BIOS and passing control to the operating system 123, via the MBR 137, and the loading and execution of the boot sector code 139 and the operating system code 133 (123). This can be seen in more detail with reference to figure 2, which is a flow diagram of a method of operation of a computer device according to an embodiment.

Following power-on of the computing device, the device's processor will look to the BIOS or other firmware module (110) for its initial instructions. The method [200] initiates a number of power-on-self-test (POST) routines [205] as would be known by those skilled in the art, but before these are completed a user log-in routine or procedure is performed [210]. This log-in procedure (300) is shown in more detail in figure 3, and allows one of a number of predetermined or pre-programmed user's to be logged in (separately and independently) or authenticated by the BIOS based login or authentication procedure [300]. Successful authentication of one of the users results in the provision of the start and end addresses of the user's partition (eg D2:/) from the user data table (UDT) – see figure 3. These addresses will typically include start and end head

(eg H2, H2'), sector (S2, S2') and cylinder (C2, C2') values for a hard disk drive (HDD) storage medium, as illustrated in figure 4.

The partition table 135 in the MBR 137 of the HDD 130 is then loaded into the working memory [215]. The partition table (PT) 125 in the RAM 120 is then modified using the user partition addresses from the UDT [220]. Typically the partition table 125 will contain one or more partitions records, one for the main or C:/ drive or partition typically containing the OS, and one for a user D:/ drive or partition, and possibly others. Each record or partition entry includes start and end addresses for the partition, its partition type and status. Typically only the main partition containing the operating system will be indicated as active and therefore bootable. The method [200] then modifies the partition table 125 in RAM by writing the start and end address entry(s) for the user partition or D:/ drive with the partition start and end addresses (eg H2, S2, C2 and H2', S2', C2') retrieved from the UDT for the authenticated user (user 2). Other parameters may be written or otherwise incorporated as required, for example the length of the partition may require changing, as may the logical block address of the first sector in the partition if used.

The file allocation table (FAT) 143 is typically located starting at the first location in the partition defined by the user partition address values (eg H2, S2, C2). The method 200 may then move on to decrypt the file allocation datastructure (D2-FAT) 143 of the user's partition (D2) 147 [225], another file within the partition (eg root directory 148), the data contents 149 of the partition, or the entire partition itself 147. Decryption of the FAT or other file allocation datastructure 143 may be achieved using any suitable mechanism available to the skilled person, and may involve the use of passwords and/or encryption algorithms from the UDT 145 and/or the firmware module 110. The method [200] then moves back to complete the remaining POST operations as normal [230], before loading and executing the master boot record (MBR) as usual [240].

In order to ensure encryption of the FAT in the case of a "brutal" shut-down of the computing device 100, the BIOS method may also include a step to encrypt all open or

decrypted FAT's prior to the user login [210]. Alternatively control of the device may pass back to the BIOS 110 prior to shut down which then encrypts all open FAT's.

Control of execution then passes away from the BIOS or firmware 110 to the MBR 137, which includes a jump instruction 132 to load and execute the master boot code 131. This locates the boot sector code 139 of the active partition (C:/) 141 of the HDD from the partition table 137, loads this into working memory 120, and calls this instantiated code 139 [245]. This boot sector 139 of the active partition (C:/) 141 includes an operating system (OS) loader which loads the operating system 133 or software program instructions from the HDD 130 into the RAM [250]. Control of the execution thread is then passed from the boot sector code 139 to the operating system 123 resident in the RAM and which is executed [255]. The operating system 123 needs to read the partition table 125 in the RAM 120 to determine the useable areas of the disk 130 and complete its boot-up process. The OS also uses the decrypted file allocation datastructure(s) (eg D2-FAT) within the HDD 130 to determine the HDD addresses for programs and/or data stored on the user partition (D2:/) as is known. The other user partitions (D1:/, D3:/) will remain invisible to the operating system, and therefore are not accessible by the currently logged on or authenticated user.

In an alternative arrangement, the PT 135 may be over-written or modified on the HDD 130 prior to it being loaded into the working memory 120 and is thus instantiated 125 already modified for use by the OS 123. As a further alternative the OS may be modified to read the modified PT 135 directly from the HDD 130.

Figures 3 and 4 show the authentication or login procedure in more detail. After the initial POST (205), the authentication method [300] generates a screen prompt for a username and password [305]. After receiving a username and password from a user [310], the user data table (UDT) or user partition datastructure 145 is decrypted using a key and decryption algorithm from within the firmware module 110 [315]. Alternatively the UDT may be retrieved from the firmware, for example from re-writable non-volatile memory such as FLASH. The UDT may also be un-encrypted in this case. The method

[300] then tries to match the entered username and password against one of the entries in the UDT 145 [320]. If no match can be found [320N], an authentication error message is displayed on the screen [325], and the method moves to re-encrypt the UDT [335]. If however a match is found [320Y], the method reads the corresponding start and end addresses to the RAM 120 area of memory [330]. A UDT 145 for multiple users is illustrated in figure 4, and comprises head, sector and cluster starting (H, S, C) and ending (H', S', C') addresses for partitions corresponding to a plurality or number of users (n). A username and password for each user is also included in this embodiment. The partition start and end addresses (eg H2, S2, C2, and H2', S2', C2') can then be used to modify the partition table (PT) in RAM by adding to any entries in the partition table as stored on th eHDD. Finally the method re-encrypts the UDT [335] so that it cannot be accessed after full boot-up of the computing device 100.

Either of the methods [200] or [300] may incorporate a restriction on the number of log-in attempts that can be made, so that for example after three failed attempts the method terminates so that the computing device must be rebooted in order to try to log-in again. Further operation of the computing device cannot continue unless the log-in has been successful, as loading of an OS cannot start.

A utility program may be provided for an administrator in order to enter, modify or delete user entries in the UDT, as would be appreciated by those skilled in the art.

The embodiment overcomes some of the shortcomings of an operating system (OS) level protection system for different partitions. Typically the OS restricts access to different disk areas using an Access Control List (ACL) or similar arrangement which spells out on the disk areas per user name. In Windows these areas can be abstracted as a folder name, and secured such that when a different user tries to access this folder the OS will prevent it. However this protection can be circumvented by avoiding loading the OS from the same HDD as the partitions during start up, for example by inserting an OS CD instead. Then the user will have full access to the entire disk without the need for passwords. The BIOS or firmware enforcement approach of the embodiment means that

OS based protection systems cannot be avoided this way, because the BIOS must be run before anything else. Thus even if the OS is loaded from an inserted CD-ROM, it will still only see the authenticated user's partition as it will still have to rely on the modified partition table passed to it by the modified BIOS.

Although the embodiment has been described with respect to a personal or laptop computer, it may also be suitable for other devices such as personal digital assistants or even shared mobile phones. Similarly, although the embodiment has been described with both a shared or main partition (C:/) and a separate user partition (D:/) which is allocated based on the user at start-up time, a single user partition may be used (ie no C:/ drive) and allocated during the BIOS routines as described above. The file allocation datastructures (eg FAT) need not be encrypted, especially where the storage media is integrated (ie not portable) with the computing device. Similarly, although the embodiment reads and writes actual address values for each user partition within the partition table or equivalent datastructure, alternative arrangements could be used to implement this same functionality.

Although the embodiment has been described with respect to Windows based datastructures such as FAT, MFT, MBR and partition tables, the skilled person will appreciate that embodiments could be implemented utilising equivalent datastructures for example in Linux or Macintosh encoded HDD or other storage media.

Figure 5 illustrates the memory architecture at a high level for a computing device such as a personal computer (PC) according to another embodiment. This arrangement is similar to that of figure 1, and common references are used for common features. The computing device 100 comprises firmware 110 such as basic input output software (BIOS) which is typically stored on read only memory (ROM) securely integrated with the device 100. The device also comprises random access memory (RAM) 120 which is typically volatile and used as the processor's working memory into which various software program instructions such as the operating system 123 and data such as a partition table 125 are loaded or instantiated and executed or modified. The device 100

also either comprises or is coupled to non-volatile storage medium or memory 130 such as a hard disk drive (HDD) which may be partitioned into a number of partitions 141, 147, for example a main or operating system partition (C:/) 141 and a user partition (D:/) 147. Each partition is self-contained as far as the operating system is concerned, and comprises its own file allocation datastructure 143, such as a file allocation table (FAT) for pre-XP Window's based systems, or a Master File Table (MFT) for an XP based system. Each partition also comprises its own boot sector code 139, root directory 148, and data area 149 for storing data and programs. In some embodiments, the root directory 148 may be combined with the file allocation datastructure 143 to create a combined file allocation and directory datastructure, for example an Inode file in Linux/Unix or a Catalog file in the Macintosh operating systems.

The BIOS or firmware instructions within the firmware module 110 load initial BIOS code into the memory 120 at a specific location and execute this, for example performing various power on self test (POST) routines.

The operating system partition (C:/) 141 comprises the operating system program code (OS) 133 which can be loaded onto the RAM 120 and executed (123), as well as a master boot record (MBR) 137 in a Window's based system. Equivalent structures exist in non-Window's based systems. The MBR 137 is the first memory location on the HDD 130 at which the system or execution thread looks after the end of the BIOS 110 start-up routines and contains a partition table 135, a master boot code 131, as well as an initial jump instruction 132 which loads the master boot code 131 into a specific location in RAM and executes this. The master boot code instantiated in RAM loads the partition table 132 into RAM (125), and interrogates the instantiated partition table 125 for the active partition (where the OS is stored and normally C:/). The boot sector code 139 for the active partition (C:/) is found and loaded into RAM at a specific location and executed. The executing boot sector code then searches the root directory 148 of the active partition for operating system files and loads and executes these. The OS then performs various operations well known to those skilled in the art to complete loading and executing of all its files.

At the end of the boot process, the OS 123 will typically execute a user or account log-in process in order to authenticate one or more predetermined users using usernames and passwords stored in a user account management file 580 or similar datastructure. Upon authentication of a user, the OS then loads certain parameters, links and/or files associated with that user as is known. For example the desktop may be customised and some user dependent start-up programs and services may be started. These start-up programs and services may be stored in the respective user's partition 147. User specific files and/or data will typically be stored in a unique partition (eg drive D:/), with any additional users being assigned additional partitions (eg E:/, F:/, and so on).

Thus for the authenticated user, there exists a unique user partition (D:/) which includes the user's data files together with a file allocation datastructure 563 and a file directory datastructure 566 for this partition (D:/). The file allocation datastructure 563 contains the storage locations such as cluster numbers or addresses of each file within the partition. The file directory 566 contains an identifier for each file together with a location or pointer to the first entry for each file in the file allocation datastructure 563. In this embodiment these file allocation and directory datastructures 563 and 566 stored on the HDD 130 are encrypted. The embodiment is arranged to instantiate the file allocation datastructure 563 of the user partition (D:/) from the HDD 130 into the RAM 120; and to decrypt this instantiated file 573. Similarly the root directory or other directory file 566 is also instantiated into the RAM 120; and decryption of this file 576 is also undertaken. In embodiments where these datastructures are combined, the combined file allocation and directory datastructure 560 is instantiated (570) into the RAM 120; and decrypted there.

The operating system is then arranged to write to (and typically read from) the file allocation datastructure 573 and the file directory datastructure 576 in the RAM, instead of the file allocation datastructure 563 and the file directory datastructure 566 in the HDD 130 as is usual. These datastructures 573 and 576 in the RAM 120 are then periodically or when amended encrypted and written back to the HDD 563 and 566. The password or key (and optionally the encryption/decryption algorithm) for the or each user can be

stored in the user account management file 580 or some other suitable file or location accessible to the OS 123.

An operating system boot-up procedure or method according to the embodiment is illustrated in figure 6, a file allocation and/or file directory datastructure read routine or method according to the embodiment is illustrated in figure 7, and a file allocation and/or file directory datastructure write routine or method according to the embodiment is illustrated in figure 8. In this embodiment, the operating system 123 calls these routines or methods instead of or in addition to the known or typical boot-up, read, and write methods in order to allow the operating system 123 to use and modify the encrypted file allocation 563 and file directory 566 datastructures on the HDD 130. These methods use and modify the instantiated and decrypted versions of these datastructures 573 and 576 in working memory 120 rather than directly on the HDD as is normal; then re-encrypt these files and write them to the HDD 130 in order to update the equivalent datastructures 563 and 566 stored there.

Referring first to figure 6, a method [600] used during the final stages of operating system boot-up is shown. This method or routine [600] can be called at any suitable point in a normal boot-up routine, for example following authentication of a user by the OS 123 but prior to loading of the users files and other parameters. This may be proceeded in a DOS system for example by loading and executing IO.SYS, MSDOS, and autoexec.bat files, and/or various other files as will be understood by those skilled in the art.

The method [600] initially encrypts the or all "open" user file allocation (563) and directory (566) datastructures [605]. This step may be included in order to ensure that the user data is secure for example following a forced shut-down of the operating system, or where in the session prior to this boot-up, one or more of the file allocation and directory datastructures 563, 566 were decrypted on the HDD 130. Typically the user account log-in will have already be performed by the OS 123, however where this has not occurred or in other circumstances a user log-in step [610] can be included in order to authenticate a user. The authenticated user will be associated with a corresponding user key which may

be stored in the user account management file 580 or another suitable file which is accessible to the OS 123. The method then reads the user key, and optionally an encryption/decryption algorithm, from the user account management file 580 or other suitable file [615]. The method then instantiates the encrypted file allocation 563 and directory 566 datastructures from the HDD 130 into the RAM 120 [620]. This simply involves copying the file from the storage medium or HDD 130 to the working memory or RAM 120. The instantiated file allocation 673 and directory 576 datastructures in the RAM 120 are then decrypted using the user key and decryption algorithm [625].

The OS 123 then continues or finishes its boot-up procedure by loading any user-specific files, services and links associated with the authenticated user. This will include reading the file allocation datastructure 573 and file directory datastructure 576 associated with the authenticated user – typically associated with a particular partition (eg D:/) - in order to identify and find the storage locations on the HDD of the various user's data files. In the embodiment, these datastructures 573 and 576 are read by the OS 123 from the working memory 120 and not the storage medium 130 as is normal. This read process is illustrated in more detail in figure 7.

Figure 7 illustrates a method of reading the contents of the file allocation datastructure 573 in order to locate a file within the storage medium. This may be combined with reading the file directory datastructure 576 where appropriate, or a combined file allocation and directory datastructure such as an Inode or Category file 570. The method [700] is called by the OS 123 each time it wishes to read the user's partition (D:/) file allocation datastructure 573 in order locate within the storage medium 130 a specified user's data file stored in this user's partition (D:/) – typically the cluster numbers used by the file. This method [700] replaces a method in which the OS calls to read the file allocation datastructure 563 directly from the HDD 130. The method [700] initially reads the decrypted file allocation datastructure 573 instantiated in RAM [705]. This datastructure contains the cluster numbers of the HDD 130 for the file to be read, in order to locate the file (contents) within the HDD. The method then locates these file contents

in the HDD [710], and reads or instantiates them into working memory or RAM [715], where they can be accessed and/or manipulated as is known.

A similar method is provided for reading the file directory datastructure 576 from the working memory 120 instead of the storage medium 130 as is usual. Typically the OS 123 will identify a requested file in the file directory datastructure 576 which contains a pointer to the first entry for that file in the file allocation datastructure 573.

Figure 8 illustrates a method of writing changes in the storage locations for a file to the file allocation datastructure 573/563 following modifications of the file in RAM 120 and writing of the modified file to the HDD 130. When the size or contents of a file are changed in working memory 120 and written to the HDD 130, or other predetermined amendments known to those skilled in the art are made to the file, the file allocation datastructure is updated with changes in where the modified file is stored in the HDD 130 as is known. In the embodiment a method [800] of performing this function using the encrypted file allocation datastructure 563 on the HDD is employed. The method initially updates the unencrypted file allocation datastructure 573 in the working memory [805], instead of directly on the HDD as normal. However the way in which the contents of this file are updated is the same as known to those skilled in the art. Thus typically a chain of clusters is included for each file, with each cluster entry including a pointer to the next cluster entry. An entry may be modified and/or a new entry added. The method then encrypts the updated file allocation datastructure 573 using the user's key, and writes this encrypted file into the HDD 130 at the usual file allocation datastructure location 563 [810]. Thus the file allocation datastructure has been updated both in working memory (573) and on the HDD (563). Then if a power failure occurs or for some other reason the computing device is "brutally" shut down, the changes to the file allocation datastructure 563 are not lost.

In an alternative arrangement, instead of updating the HDD version of the encrypted file allocation datastructure 563 after every change to the file allocation datastructure 573 instantiated in working memory 120, this may be carried out periodically.

The method [800] then determines whether any change to the file directory datastructure 576/566 is required [815]. This may be required where the user file to be written to the HDD has had its name changed, or some other parameter which will be known to those skilled in the art. If a directory change or amendment is required [815Y], then the file directory datastructure 576 in the working memory is updated [820]. However the way in which the contents of this file are updated is the same as known to those skilled in the art. The method then encrypts the updated file directory datastructure 576 using the user's key, and writes this encrypted file into the HDD 130 at the usual file directory datastructure location 566 [825]. Thus the file directory datastructure has been updated both in working memory (576) and on the HDD (566). Then if a power failure occurs or for some other reason the computing device is "brutally" shut down, the changes to the file directory datastructure 563 are not lost.

A similar method of updating a single file comprising the file allocation and directory information can be performed in an analogous manner, by first updating the file in memory 120, then encrypting it and writing the encrypted file back to its original location on the HDD.

A utility program may also be provided for an administrator in order to enter, modify or delete user entries in the UA or other file, including keys for each user, as would be appreciated by those skilled in the art.

The embodiment may additionally comprise a user data table (UDT) 145 or other user partition datastructure which contains the start and end addresses (head-sector-cylinder) for a number of user partitions (D1:/ - D3:/) 147 as previously described. The partition table will have an entry added depending on user log-in. This arrangement combined with the encrypted file allocation and/or directory datastructures further enhances the security of user data on a HDD or other storage media partition.

Figure 9 illustrates another embodiment which utilises a modified BIOS as well as a modified operating system in order to implement secure user data features. Following power-on of the computing device, the device's processor will look to the BIOS or other firmware module (110) for its initial instructions. These are loaded into the working memory or RAM 120, and the method [900] initiates a number of power-on-self-test (POST) routines [905] as would be known by those skilled in the art. However before these are completed the method encrypts all open user file allocation and directory datastructures 563 and 566 as previously described [910]. A user log-in routine or procedure is then performed [915]. This log-in procedure may be similar to that of figure 3, but providing a decryption key instead of or in addition to user partition table entries. This allows one or more predetermined or pre-programmed user's to be logged in (separately and independently) or authenticated by the BIOS based login or authentication procedure. Successful authentication of one of the users results in the provision of the a user key for encryption/decryption of the file allocation and/or directory datastructures 563/573 and 566/576 associated with the user's partition (eg D:/). The key is initially written to a specified location in the working memory or RAM 120 for use by the method. The BIOS is then arranged to load the partition table from the HDD to the RAM as known [920].

Where the user partition start and end addresses are provided following successful user authentication, the instantiated storage medium partition datastructure 125 is modified as previously described [925].

The method then completes the POST [930], before moving to the first sector on the HDD 130 to load the master boot code 131 from the master boot record 137 into RAM 120, and to execute this [935]. Control of the computing device then passes from the BIOS to the MBR 137. The master boot code 131 locates the active partition from the partition table 135 and loads the boot sector code 139 of the active partition into RAM 120, and executes this [940]. Control of the computing device then passes to the boot sector code 139, which finds and loads the (or some of the) operating system files 133 into RAM 130 [945]. The loaded OS files 123 are executed and continue the boot-up

process. Control of the computing device is thus passed to the OS 123, which continues its boot-up process and performs another OS level user login procedure [950]. This may be performed as described with respect to figure 6 in order to provide the decryption key. Alterntively the decryption key may be provided as part of the BIOS level user login as described above [915].

The operating system then continues as is known, for example by reading the file allocation (eg FAT) 143 and directory (eg root directory) 148 datastructures or files for the active partition directly from the HDD 130 as is known [955]. The method then reads the (encrypted) file allocation 563 and directory 566 datastructures for the user's partition (D:/) from the HDD 130 to the working memory 120 [960]. Finally the method decrypts the file allocation and directory datastructures using the user key [965]. These decrypted files or datastructures 573 and 576 are then used by the OS as previously described with respect to figures 7 and 8, to read/write changes and if required encrypt the changed files and write them back to the HDD.

The method [900] may incorporate a restriction on the number of log-in attempts that can be made, so that for example after three failed attempts the method terminates so that the computing device must be rebooted in order to try to log-in again. Further operation of the computing device cannot continue unless the log-in has been successful, as reading of the user's file allocation datastructure 563 cannot start.

This embodiment provides two levels of encryption: first or initially at the BIOS level where the partition table may be modified, followed by the second or further operating system level, where the FAT and directory structures are decrypted. Failing either one of the two logins prevents a malicious user from accessing the HDD contents.

The skilled person will recognise that the above-described apparatus and methods may be embodied as processor control code, for example on a carrier medium such as a disk, CD- or DVD-ROM, programmed memory such as read only memory (Firmware), or on a data carrier such as an optical or electrical signal carrier. For many applications embodiments

of the invention will be implemented on a DSP (Digital Signal Processor), ASIC (Application Specific Integrated Circuit) or FPGA (Field Programmable Gate Array). Thus the code may comprise conventional programme code or microcode or, for example code for setting up or controlling an ASIC or FPGA. The code may also comprise code for dynamically configuring re-configurable apparatus such as re-programmable logic gate arrays. Similarly the code may comprise code for a hardware description language such as Verilog ™ or VHDL (Very high speed integrated circuit Hardware Description Language). As the skilled person will appreciate, the code may be distributed between a plurality of coupled components in communication with one another. Where appropriate, the embodiments may also be implemented using code running on a field-(re)programmable analogue array or similar device in order to configure analogue hardware.

The skilled person will also appreciate that the various embodiments and specific features described with respect to them could be freely combined with the other embodiments or their specifically described features in general accordance with the above teaching. The skilled person will also recognise that various alterations and modifications can be made to specific examples described without departing from the scope of the appended claims.

CLAIMS

1. A method of operating a computing device having working memory and coupled to a storage medium, the method comprising:

instantiating an encrypted file allocation datastructure stored on the storage medium into the working memory, the file allocation datastructure comprising storage locations on the storage medium for a number of files;

decrypting the file allocation datastructure in working memory;

reading the decrypted file allocation datastructure in order to locate a said file within the storage media.

2. A method according to claim 1, wherein the storage medium comprises a plurality of user partitions each associated one of a plurality of users and each comprising a respective encrypted file allocation datastructure, the method arranged to instantiate and decrypt only one of the encrypted file allocation datastructures in response to authentication of one of the plurality of users.

3. A method according to claim 1 or 2, wherein the method is performed by an operating system executing on the computing device as part of a boot-up procedure such that access the files associated with an encrypted file allocation datastructure which has not been decrypted by the method is prevented by the operating system.

4. A method according to any one preceding claim, further comprising amending an entry associated with the file in the file allocation datastructure in the working memory in response to predetermined amendments to the file on the storage medium.

5. A method according to claim 4, further comprising encrypting and writing said amended file allocation datastructure to said storage medium in response to said predetermined amendments.

6. A method according to any one preceding claim, further comprising:

performing a user login procedure in order to authenticate a predetermined user;

performing said encrypted file allocation datastructure instantiation only in response to authentication of said predetermined user.

7. A method according to claim 6, further comprising encrypting the file allocation datastructure in the storage medium prior to performing said login procedure.

8. A method according to claim 6 or 7, wherein the login procedure comprises an initial login procedure performed by firmware program instructions stored in a firmware module securely integrated into the computing device; and a further login procedure performed by operating system instructions stored on the storage medium.

9. A method according to any one preceding claim, further comprising:

executing firmware program instructions associated with the computing device on powering on of the computing device, and which executing firmware program instructions perform the following: authenticate one of a plurality of users; determine a user partition start and end address for the authenticated user from a user partition datastructure which comprises user partition start and end addresses on the storage medium for the plurality of users, and incorporate the determined user partition start and end addresses into a storage medium partition datastructure associated with the storage medium;

loading and executing operating system program instructions for operating the computing device from the storage medium following incorporating of the determined user partition start and end addresses into the storage medium partition datastructure, and which operating system program instructions require the storage medium partition datastructure for said execution.

10. A method according to any one preceding claim further comprising instantiating an encrypted file directory datastructure stored on the storage medium into the working memory, the file directory datastructure comprising directions to an entry in the file allocation datastructure for a number of files;

decrypting the file directory datastructure in working memory;

reading the decrypted file directory datastructure in order to locate a said file entry within the file allocation datastructure.

11. A carrier medium carrying processor code which when executed on a processor causes the processor to carry out the method of any one preceding claim.

12. A computing device for coupling to a storage medium, the device comprising:

a processor arranged to load and execute operating system program instructions stored on the or another storage medium, the operating system program instructions when executed on the processor arranged to instantiate an encrypted file allocation datastructure stored on the storage medium into a working memory, the file allocation datastructure comprising storage locations on the storage medium for a number of files;

the operating system program instructions further arranged to decrypt the file allocation datastructure in working memory, and read the decrypted file allocation datastructure in order to locate a said file within the storage media.

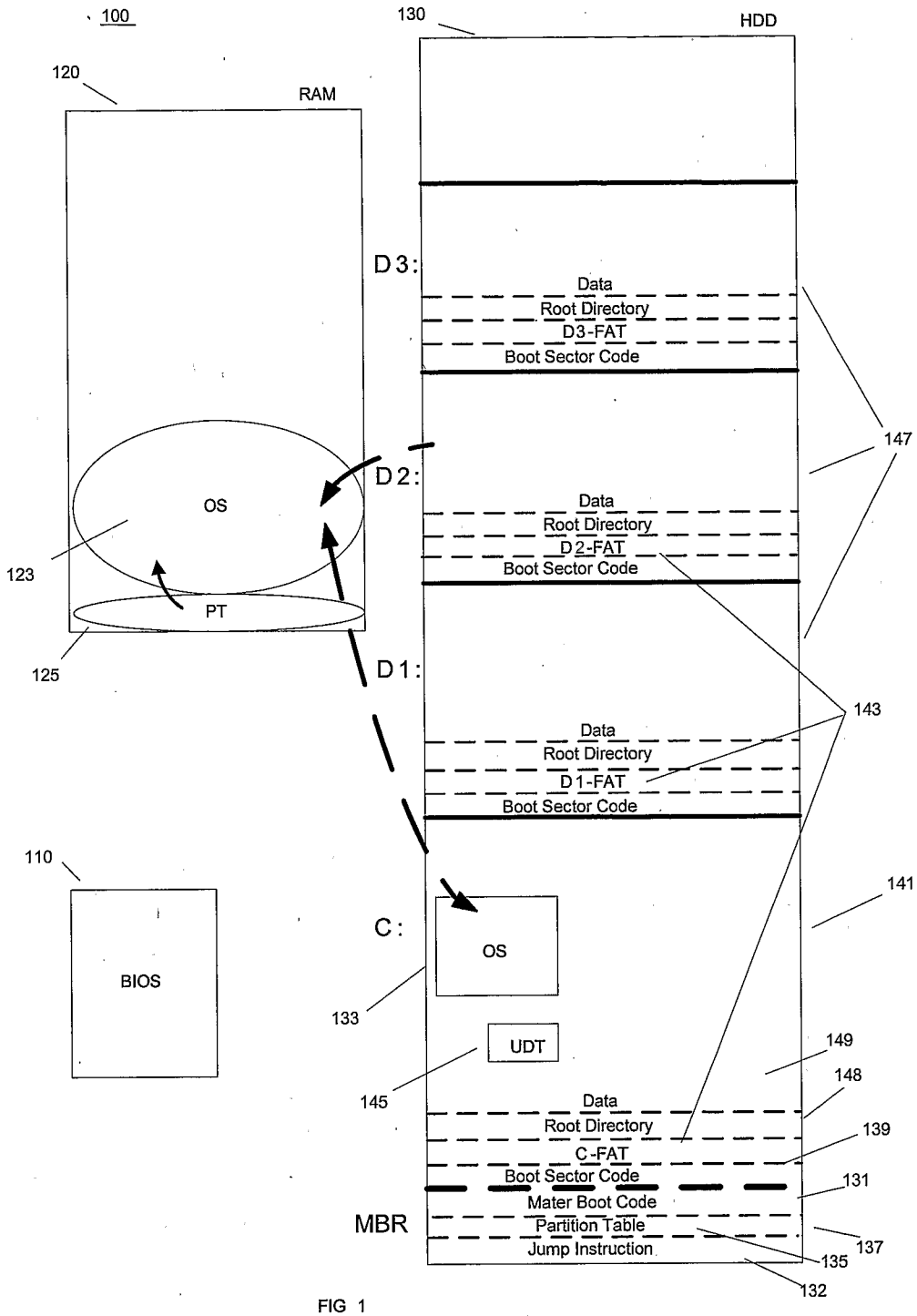

100
130
HDD
120
RAM
D3:
Data
Root Directory
D3-FAT
Boot Sector Code
147
D2:
OS
Data
Root Directory
D2-FAT
Boot Sector Code
123
PT
125
D1:
143
Data
Root Directory
D1-FAT
Boot Sector Code
110
141
C:
OS
BIOS
133
UDT
149
148
145
Data
Root Directory
C-FAT
139
Boot Sector Code
131
Mater Boot Code
MBR
Partition Table
137
Jump Instruction
135
132

FIG 1

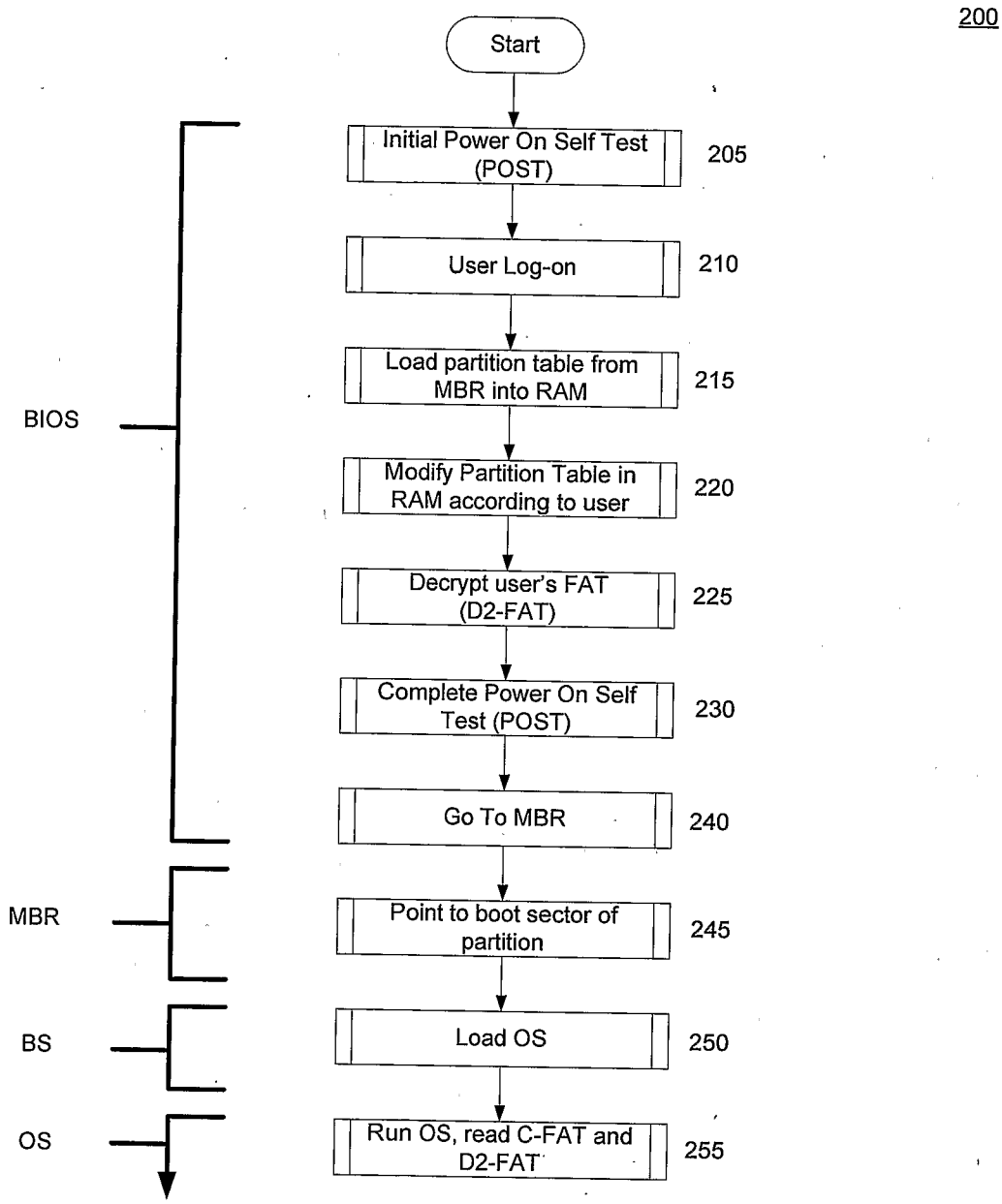

200
Start
Initial Power On Self Test (POST)
205
User Log-on
210
Load partition table from MBR into RAM
215
Modify Partition Table in RAM according to user
220
Decrypt user's FAT (D2-FAT)
225
Complete Power On Self Test (POST)
230
Go To MBR
240
Point to boot sector of partition
245
Load OS
250
Run OS, read C-FAT and D2-FAT
255
BIOS
MBR
BS
OS

FIG 2

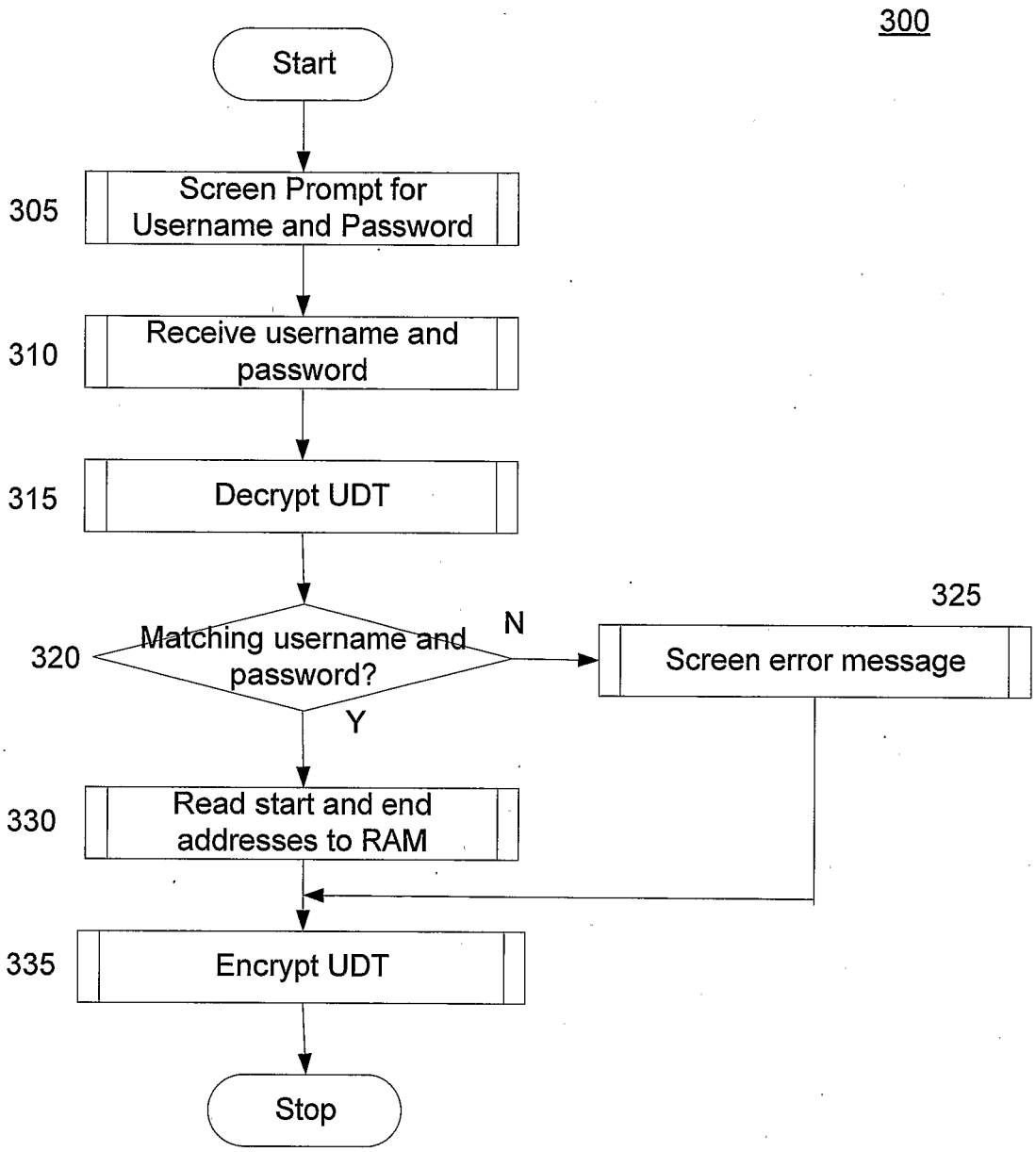

300
Start
305
Screen Prompt for Username and Password
310
Receive username and password
315
Decrypt UDT
320
Matching username and password?
N
325
Screen error message
Y
330
Read start and end addresses to RAM
335
Encrypt UDT
Stop

FIG 3

Encrypted User Data Table (UDT)

Username	Password	Starting Partition			Ending Partition		
		Head	Sector	Cylinder	Head	Sector	Cylinder
User 1	Pass 1	H1	S1	C1	H1'	S1'	C1'
User 2	Pass 2	H2	S2	C2	H2'	S2'	C2'
User *n*	Pass *n*	H*n*	S*n*	C*n*	H*n'*	S*n'*	C*n'*

FIG 4

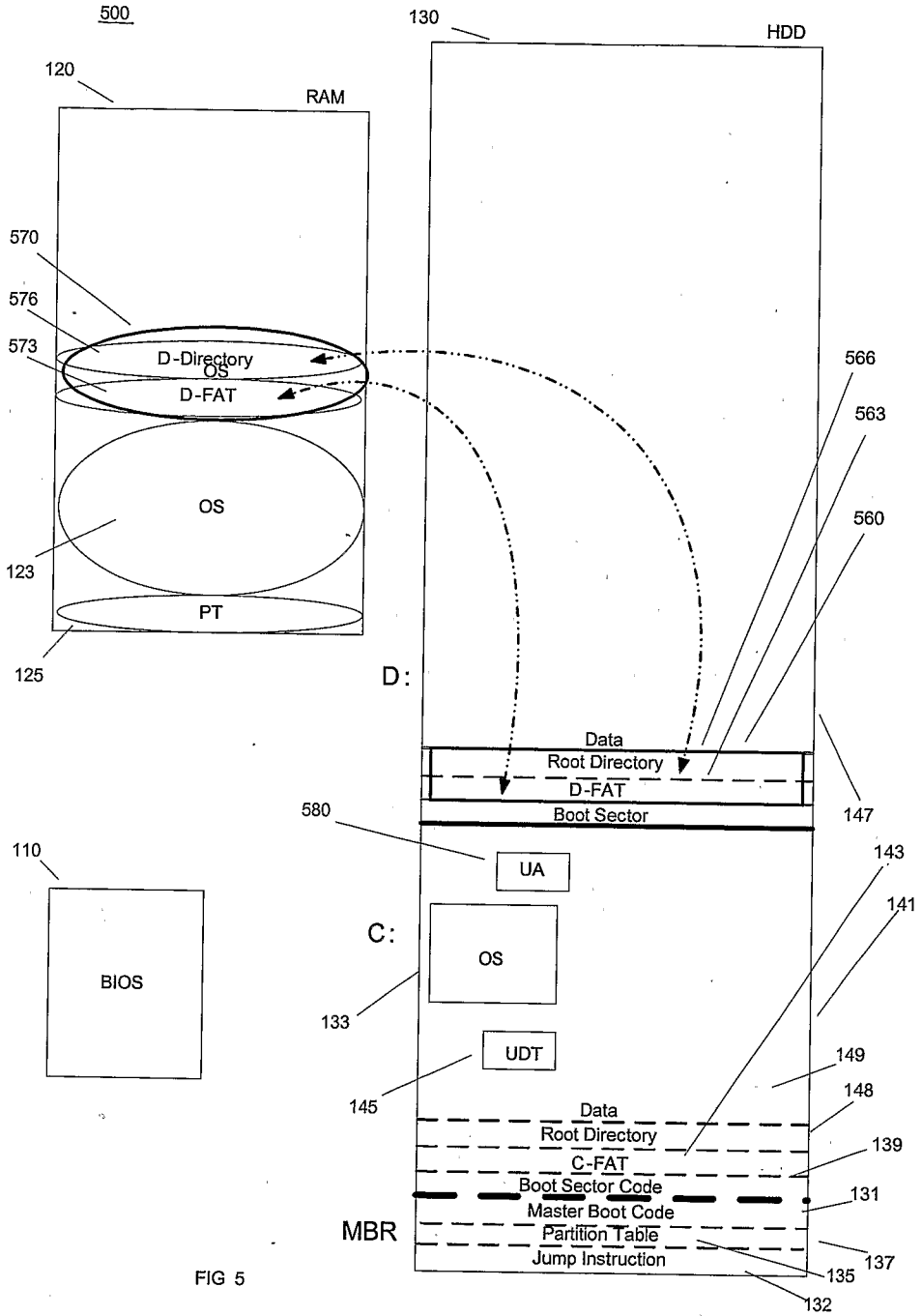

500
130
HDD
120
RAM
570
576
573
D-Directory
OS
D-FAT
OS
123
PT
125
566
563
560
D:
Data
Root Directory
D-FAT
Boot Sector
147
580
UA
110
BIOS
C:
OS
133
UDT
145
143
141
149
148
Data
Root Directory
C-FAT
139
Boot Sector Code
131
Master Boot Code
MBR
Partition Table
Jump Instruction
135
137
132

FIG 5

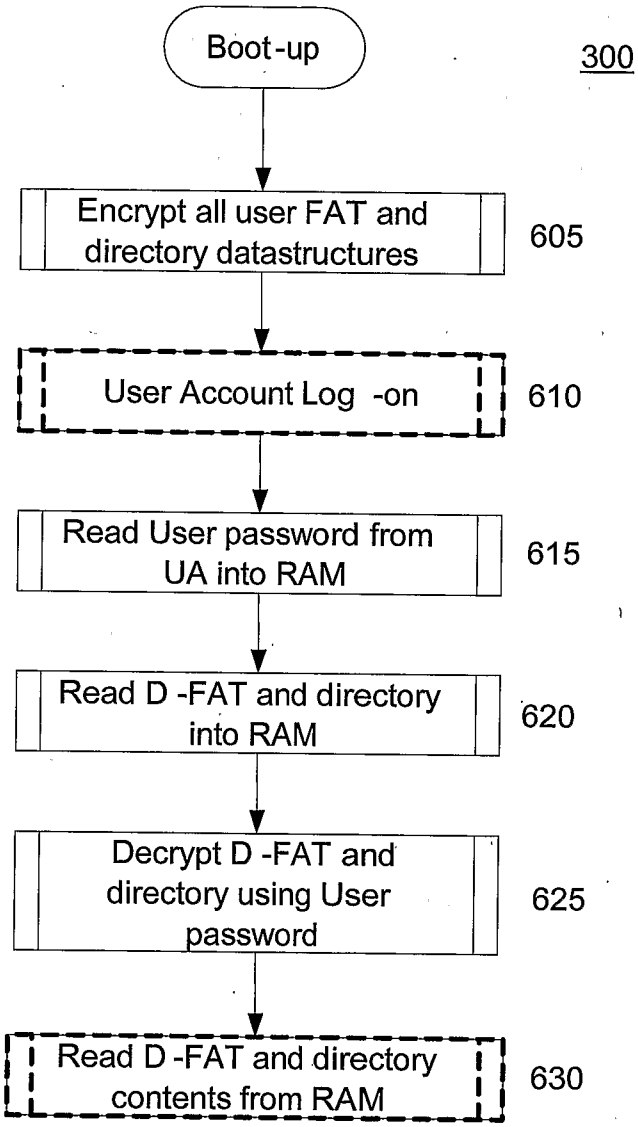

Boot-up
300
Encrypt all user FAT and directory datastructures
605
User Account Log -on
610
Read User password from UA into RAM
615
Read D -FAT and directory into RAM
620
Decrypt D -FAT and directory using User password
625
Read D -FAT and directory contents from RAM
630

FIG 6

700

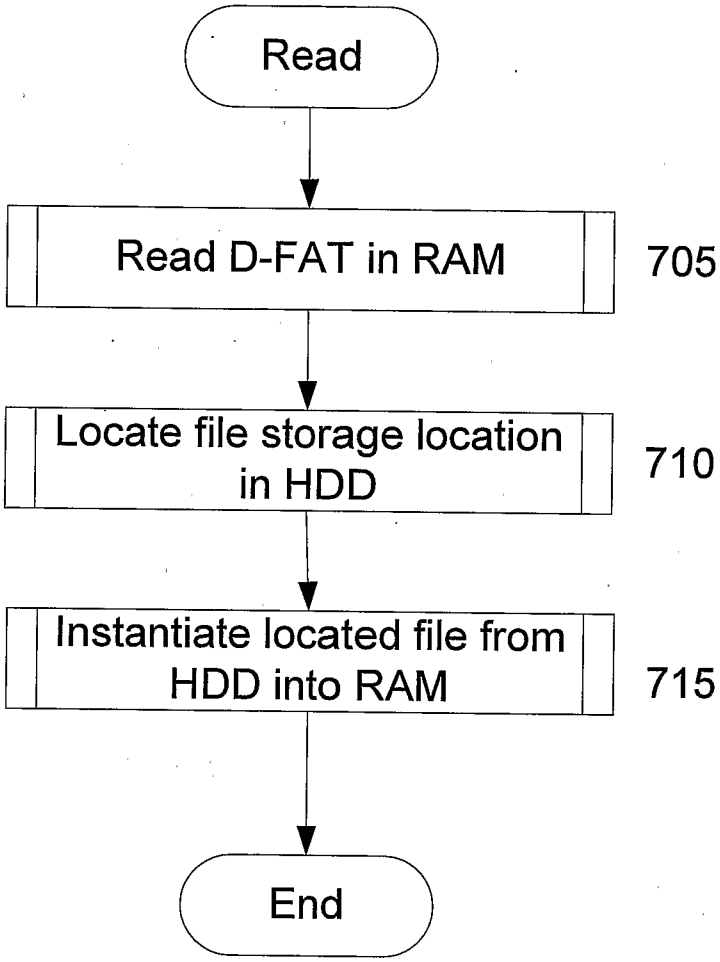

Read
Read D-FAT in RAM
705
Locate file storage location in HDD
710
Instantiate located file from HDD into RAM
715
End

FIG 7

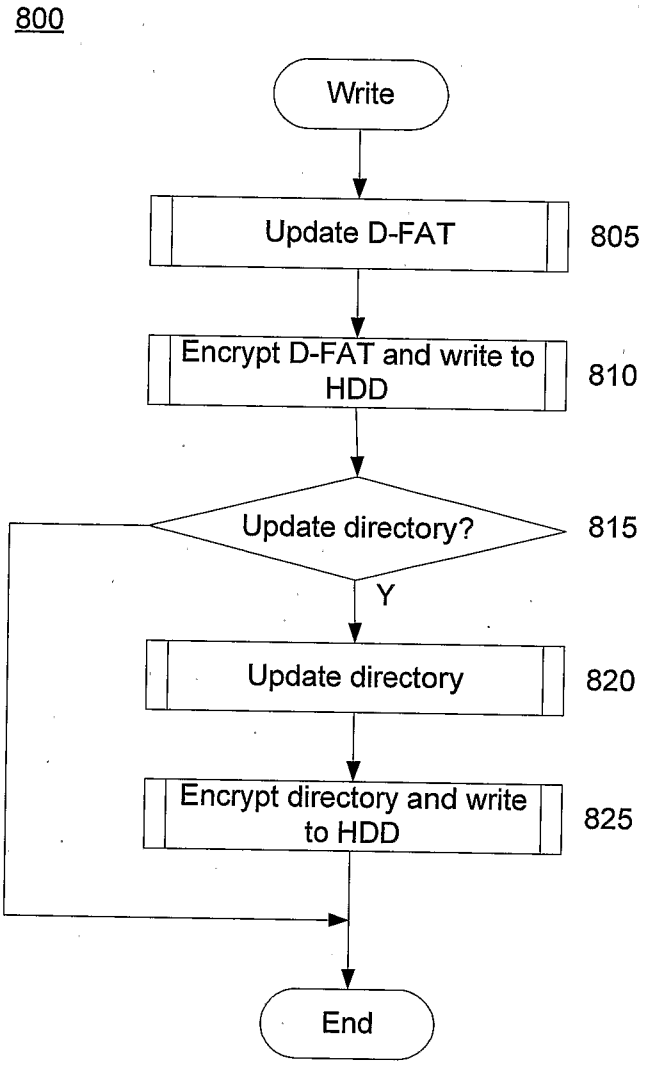

800
Write
Update D-FAT
805
Encrypt D-FAT and write to HDD
810
Update directory?
815
Y
Update directory
820
Encrypt directory and write to HDD
825
End

FIG 8

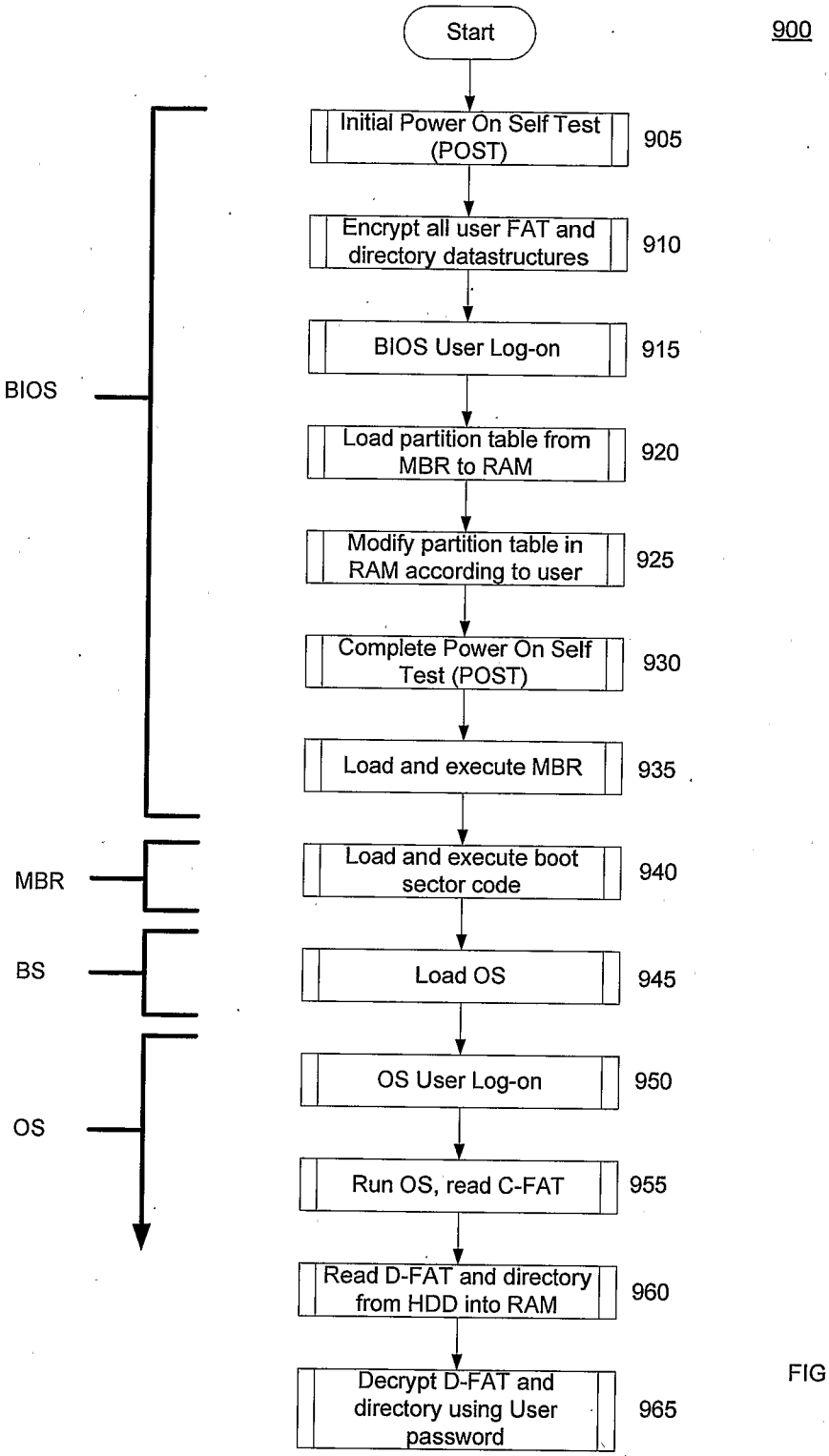

Start
900
Initial Power On Self Test (POST)
905
Encrypt all user FAT and directory datastructures
910
BIOS User Log-on
915
Load partition table from MBR to RAM
920
Modify partition table in RAM according to user
925
Complete Power On Self Test (POST)
930
Load and execute MBR
935
Load and execute boot sector code
940
Load OS
945
OS User Log-on
950
Run OS, read C-FAT
955
Read D-FAT and directory from HDD into RAM
960
Decrypt D-FAT and directory using User password
965
BIOS
MBR
BS
OS

FIG 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/GB2007/002900

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/02

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

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/158711 A1 (ZIMMER VINCENT J [US]) 12 August 2004 (2004-08-12) abstract; figures 1,2 paragraph [0052] paragraph [0054] - paragraph [0056] paragraph [0059] paragraph [0066] paragraph [0069] - paragraph [0070] paragraph [0072]	1-12
A	US 6 374 266 B1 (SHNELVAR RALPH [US]) 16 April 2002 (2002-04-16)	
A	US 2005/193195 A1 (WU KUEN-TSAN [TW] ET AL) 1 September 2005 (2005-09-01)	
A	WO 2006/069274 A (SANDISK CORP [US]; JAGOND-COULOMB FABRICE [US]; HOLTZMAN MICHAEL [US];) 29 June 2006 (2006-06-29)	

☐ 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 document but published on or after the international filing date

"L" 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 principle 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
19 October 2007	29/10/2007

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Powell, David

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/GB2007/002900

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004158711 A1	12-08-2004	US 2007061562 A1	15-03-2007
US 6374266 B1	16-04-2002	NONE	
US 2005193195 A1	01-09-2005	NONE	
WO 2006069274 A	29-06-2006	EP 1836641 A2	26-09-2007