IN-PLAY DETECTION OF ALTERED GAME DATA

Identify Select Game Data/Addresses

Compute Cryptographic Hash Values For Select Game Data/Addresses

Store Select Game Data in Memory

Generate Obfuscated Lookup Table

Protect Selected Memory Locations

Receive Challenge

Locate Selected Memory Locations Via Obfuscated Lookup Table

Inspect Select Data

Compute Cryptographic Hash Values

Provide Cryptographic Hash Values

Compare Received Hash Values With Expected Hash Values

Match?

Continue Online Game

Halt Online Game

Correspondence Address:
WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR, 2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

ABSTRACT

An online service provides detection of tampering of game data. During game play, the service provides challenges to inspect select data in memory of a game device. A challenge includes an executable program that is configured to locate select data and analyze the select data for tampering. Upon locating the select data, the challenge computes cryptographic hash values from the located select data and returns those hash values to the online game service, where they are compared against expected hash values to determine whether data tampering has occurred on the game device. If the cryptographic hash values match, the service allows online gaming to continue. If the cryptographic hash values do not match, the service discontinues online gaming by terminating the game session for example.
FIGURE 1

12 Identify Select Game Data/Addresses
14 Compute Cryptographic Hash Values For Select Game Data/Addresses
16 Store Select Game Data In Memory
18 Generate Obfuscated Lookup Table
20 Protect Selected Memory Locations
22 Receive Challenge
24 Locate Selected Memory Locations Via Obfuscated Lookup Table
26 Inspect Select Data
26 Compute Cryptographic Hash Values
27 Provide Cryptographic Hash Values
28 Compare Received Hash Values With Expected Hash Values
32 Match? (branch)
34 Continue Online Game
35 Halt Online Game
FIGURE 3
IN-PLAY DETECTION OF ALTERED GAME DATA

TECHNICAL FIELD

[0001] The technical field relates generally to computer processing and more specifically to online gaming.

BACKGROUND

[0002] It is not uncommon for online game players to cheat in order to appear to be better players than they truly are. Cheating can adversely affect online game communities and can significantly impact a player's desire to play against others online. Players are known to cheat via utilities that modify a game's data memory at runtime. Modifications can include, for example, changes to game data constants and/or characteristics, such as the amount of ammunition, the strength of an item, the health of a player, the position of walls, deleting of walls from a map to enable a player to shoot through walls in the game, or the like. Modifications are commonly encapsulated in small cheat applications colloquially called "Trainers." Because typical file tampering mechanisms verify the integrity of a file on-disk (e.g., by verifying a digital signature), once a file is loaded into memory, it can be modified without affecting the file on-disk (e.g., the file on-disk remains valid, although its in-memory representation has been altered). Thus, trainers can be applied directly to a game's memory during play, and avoid detection by file tampering detection mechanisms implemented by the game.

SUMMARY

[0003] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description Of Illustrative Embodiments. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

[0004] Tampering of select game data is detectable during game play. During game play, challenges to inspect select data are provided to a game device, such as a game console or the like. Memory locations of the game device in which the select data are stored are analyzed to determine if the data has been altered. If data has been altered, online execution of the game ceases (e.g., login session terminated). If data has not been altered, online game play continues. In an example embodiment, the challenges include references to the select data. The challenge, upon receipt by the game device, locates the select data in game device memory, and computes cryptographic hash values from the select data stored in the game device memory. In response to the challenge, the game device provides the cryptographic hash values of the select data. Upon receipt of the hash values by the challenger (e.g., by a server), the received cryptographic hash values are compared with expected hash values. If the cryptographic hash values match, online game execution continues. If the cryptographic hash values do not match, game execution is halted.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The foregoing summary, as well as the following detailed description, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating in-play detection of altered game data, there is shown in the drawings exemplary constructions thereof, however, in-play detection of altered game data is not limited to the specific methods and instrumentalities disclosed.

[0006] FIG. 1 is a flow diagram of an example process for providing in-play detection of altered game data.

[0007] FIG. 2 is a flow diagram of an exemplary processor for implementing in-play detection of altered game data.

[0008] FIG. 3 is an illustration of functional components of a multimedia/gaming console that can be used to implement in-play detection of altered game data.

[0009] FIG. 4 is a depiction of a suitable computing environment in which in-play detection of altered game data can be implemented.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0010] During execution of a game, altered game data is detectable via challenges provided to a game device. A challenge includes an executable program that is capable of locating select data in memory and analyzing the select data to determine if the select data has been altered. In an example embodiment, select data are stored in memory of the game device. When a challenge is received by the game device, the challenge is loaded and the executable program therein is executed. The challenge locates, via an obfuscated lookup table, the select data stored in the memory of the game device. The challenge obtains the select data from the memory of the game device and computes cryptographic hash values from the obtained data. The calculated hash values are compared with hash values provided as part of the challenge. If the hash values match, game execution continues. If the hash values do not match, game execution is halted.

[0011] FIG. 1 is a flow diagram of an example process for in-play detection of altered game data. The process depicted in FIG. 1 is described herein in the context of a game executing on a game console (e.g., XBOX® game console) utilized in an online gaming (XBOX® LIVE) scenario. It is to be understood that this context is exemplary and applications of in-play detection of altered game data should not be limited thereto.

[0012] Select game data to be analyzed for alteration is identified at step 12. Select game data can be any appropriate data such as game constants, static characteristics, attributes, or the like. For example, select game data can include the amount of ammunition (e.g., bullets) assigned to a player, the number of walls in a game scenario, the ability to shoot through walls, the maximum health of a player, the maximum strength of a player, the maximum life/duration of a player, or the like. The relative addresses of the select game data also are identified at step 12. The select data and the relative addresses of the select data can be identified, for example, by a developer of the game. For example, the developer can establish a name for each data segment of the select data. And, the names of the data segments can be used during the challenge to map the requested data segments to entries in an asset table indicating the actual in-memory locations of the data segments.

[0013] Cryptographic hash values associated with the select game data and/or the relative addresses are generated at step 14. A cryptographic hash value is the value obtained from performing a hash function. A cryptographic hash function is a function that converts a variable length input into a fixed length output, referred to as the hash value. Within mathematical limits, two different inputs to a hash function will not result in the same hash value. In an exemplary embodiment, a cryptographic hash function, such as the well known MD5,
SHA-1, or SHA-256, for example, is used to obtain hash values for the select data. Cryptographic hash values can be generated for any appropriate portions of the select data, such as, for example, static game data. For example, assume a particular map texture is 5 Kbytes in size. This could be hashed down to 20 bytes, and 20-byte hash value would be returned to the server in response to a challenge asking for the hash of that map texture. If the 20 bytes returned by the client match the 20 bytes that the server knows to be the correct hash value, the challenge has been successfully responded to and the client's login session is not terminated. In an example embodiment, an XML file of the names and relative addresses of the select data are generated at step 14.

[0014] The select data game data is stored in the game device at step 16. The game data can be stored in any appropriate location, such as storage in the game device and/or storage external to the game device. In an example embodiment, the select data game data is stored in memory in the game device. The select data, along with other game information is loaded into the game device in order to execute the game. In an example embodiment, the game information, including the select data, is stored on a disc (e.g., optical disc) and provided to a user (also referred to as a player). The player inserts the disc into a game device to play a game. The select game data is stored in random locations in memory in the game device. At step 18, a lookup table is generated indicating the locations in which the select data have been stored. The lookup table is obfuscated. The lookup table can be obfuscated in any appropriate manner, for example the lookup table can be AES-encrypted (e.g., encrypted in accordance with the Advanced Encryption Standard) using a symmetric key stored securely within the program, in secure hardware, of the like. Keys can be securely stored in any appropriate manner. For example, keys can be securely stored in hardware, such as a TPM (Trusted Platform Module) chip or the like, and/or keys can be obfuscated in any appropriate manner. Encryption prevents attackers from reading the data in “clear text.” Further, a hash/signature can be utilized to ensure the integrity of the table. Also, an nonce (a value used once only) can be used to prevent replay attacks in which a legitimate (encrypted) table is copied from one process and later injected into another process. In an example embodiment, the game calls an Application Programming Interface (API) that provides the base memory locations in which the select data are stored, the size of the memory segments in which the select data are stored, and the appropriate name of the data segment, as established at step 12. The API generates the obfuscated lookup table and registers the memory for inspection. As described above, once a data segment has been assigned a name, the data segment can be referenced by a subsequent challenge, which will result in the associated memory, in which being hashed, with the resultant hash being returned as a response to the challenge. The memory in which the select data is stored is protected at step 20. In an example embodiment, the memory in which the select data is stored is designated as read-only memory to prevent tampering and/or any inadvertent modification to the select data.

[0015] During game execution, while signed in to an online game service, such as XBOX® LIVE for example, a challenge is received by the game device at step 22. A challenge can be provided by any appropriate source. For example, challenges can be provided by an online game service (e.g., XBOX® LIVE). A challenge can be received at any time during game play. For example, challenges can be received randomly, periodically, or a combination thereof. The challenge comprises an executable program configured to inspect the select data for an indication of alteration. The challenge can be in any appropriate form. For example, the challenge can be in the form of a module comprising a Dll and a data manifest, such as an XML file or the like. In an example embodiment, the data manifest includes an XML file that comprises indications (e.g., names) of portions of the select data. In an example embodiment, the XML file comprises a list of data segment names for which hash values are to be calculated and returned (assuming the requested data segment is presently loaded/registered in the process). In response to the challenge being received, the executable program in the challenge is executed. At step 24, the challenge locates the select data stored in the game device memory via the obfuscated lookup table. The challenge accesses the select data stored in memory and operates (at step 26) on the selected data with hash functions to obtain hash values indicative of the selected data stored in the game device memory. In an example embodiment, the data segments are located in memory and a hash value is calculated for each data segment. In response to the challenge, an indication of the selected data in memory is provided at step 27. In an example embodiment, the indication of the select data comprises the calculated hash values. In an example embodiment, the calculated hash values are provided to the challenger (e.g., server of the online game service). The calculated hash values are compared, at step 28, with expected hash values. Hash values may not match for example, if the select data stored in the game memory was altered. Hash values of altered select data, within mathematical limits, will differ from hash values of unaltered select data. In an example embodiment, the comparisons of hash values are performed on a server of an online game service. If, at step 32, the expected hash values match the hash values calculated from the select data stored in game memory were received as part of the response to the challenge, online game execution is allowed to continue at step 34. If, at step 32, the hash values do not match, online game execution is halted at step 35. Online game execution can be halted in any appropriate manner, such as terminating the game device login session for example. In various embodiments, a user can be barred from online gaming for a period of time, if it is determined that the select data in memory has been altered.

[0016] FIG. 2 is a diagram of an exemplary processor 36 for implementing in-play detection of altered game data. In an example embodiment, the processor 36 comprises the game device that can be utilized to achieve online gaming. In this example embodiment, the game device can log on to a game service. During game play, if the game service detects tempering of select data stored in memory of the game device, the game service can discontinue online game play by disconnecting the game device from the online game service. In an example configuration, the game service comprises at least one server that can provide a challenge to the game device, receive hash values calculated in accordance with select data in memory of the game device, and compare the receive hash values with expected hash values.

[0017] The processor 36 comprises a processing portion 38, a memory portion 40, and an input/output portion 42. The processing portion 38, memory portion 40, and input/output portion 42 are coupled together (coupling not shown in FIG. 2) to allow communications therebetween. The input/output portion 42 is capable of providing and/or receiving compo-
ments utilized to implement in-play detection of altered game data as described above. For example, the input/output portion 42 is capable of providing and/or receiving the select data, hash values associated with the select data, a challenge, or a combination thereof.

[0018] The processing portion 38 is capable of implementing in-play detection of altered game data as described above. For example, the processing portion 38 is capable of storing select data in memory portion 40, generating a lookup table, obfuscating the lookup table, protecting stored select data, locating select data in memory via the lookup table, loading an executable program of the challenge, executing the executable program of the challenge, calculating hash values, providing hash values to the input/output portion 42, or a combination thereof.

[0019] The processor 36 can be implemented as a client processor and/or a server processor. In a basic configuration, the processor 36 can include at least one processing portion 38 and memory portion 40. The memory portion 40 can store any information utilized in conjunction with in-play detection of altered game data. For example, the memory portion 40 can store the select data, the lookup table, hash values, names of portions of the select data, or a combination thereof. Depending upon the exact configuration and type of processor, the memory portion 40 can be volatile (such as RAM) 44, non-volatile (such as ROM, flash memory, etc.) 46, or a combination thereof. The processor 36 can have additional features or functionality. For example, the processor 36 can include additional storage (removable storage 48 and/or non-removable storage 50) including, but not limited to, magnetic or optical disks, tape, flash, smart cards or a combination thereof. Computer storage media, such as memory portion 40, 44, 46, 48, and 50, include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media include, but are not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, universal serial bus (USB) compatible memory, smart cards, or any other medium which can be used to store the desired information and which can be accessed by the processor 36. Any such computer storage media can be a part of the processor 36.

[0020] The processor 36 can also contain communications connection(s) 56 that allow the processor 36 to communicate with other devices, such as other devices in an online gaming scenario, for example. Communications connection(s) 56 is an example of a communication media. Communication media typically embody computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term “modulated data signal” means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media. The processor 36 also can have input device(s) 54 such as a keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 52 such as a display, speakers, printer, etc. also can be included. In an example embodiment, output device 52 comprises display portion 28.

[0021] FIG. 3 illustrates functional components of a multimedia gaming console 300 that can be used to implement in-play detection of altered game data. In an example embodiment, the multimedia console 300 represents a more detailed depiction of a game device, such as the processor 36 implemented as a game device. In this example embodiment, the memory portions, processing portions, and the input/output portions of the multimedia console 300 are capable of performing the functions of the memory portions, processing portions, and the input/output portions of the processor 36, respectively. The multimedia console 300 has a central processing unit (CPU) 301 having a level 3 cache 302, a level 2 cache 304, and a flash ROM (Read Only Memory) 306. The level 3 cache 302 and a level 2 cache 304 temporarily store data and hence reduce the number of memory access cycles, thereby improving processing speed and throughput. The CPU 301 can be provided having more than one core, and thus, additional level 3 and level 2 caches 302 and 304. The flash ROM 306 can store executable code that is loaded during an initial phase of a boot process when the multimedia console 300 is powered ON.

[0022] A graphics processing unit (GPU) 308 and a video encoder/video codec (coder/decoder) 314 form a video processing pipeline for high speed and high resolution graphics processing. Data is carried from the graphics processing unit 308 to the video encoder/video codec 314 via a bus. The video processing pipeline outputs data to an A/V (audio/video) port 340 for transmission to a television or other display. A memory controller 310 is connected to the GPU 308 to facilitate processor access to various types of memory 312, such as, but not limited to, a RAM (Random Access Memory).

[0023] In an exemplary embodiment, the multimedia console 300 includes an input/output (I/O) controller 320, a system management controller 322, an audio processing unit 323, a network interface controller 324, a first USB host controller 326, a second USB controller 328 and a front panel I/O subassembly 330 that can be implemented on a module 318. The USB controllers 326 and 328 serve as hosts for peripheral controllers 342(1)-142(2), a wireless adapter 348, and an external memory device 346 (e.g., flash memory, external CD/DVD ROM drive, removable media, etc.). The network interface 324 and/or wireless adapter 348 provide access to a network (e.g., the Internet, home network, etc.) and can be any of a wide variety of various wired or wireless adapter components including an Ethernet card, a modem, a Bluetooth module, a cable modem, and the like.

[0024] System memory 343 is provided to store application data that is loaded during the boot process. A media drive 344 is provided and can comprise a DVD/CD drive, hard drive, or other removable media drive, etc. The media drive 344 can be internal or external to the multimedia console 300. Application data can be accessed via the media drive 344 for execution, playback, etc. by the multimedia console 300. The media drive 344 is connected to the I/O controller 320 via a bus, such as a Serial ATA bus or other high speed connection (e.g., IEEE 3394).

[0025] The system management controller 322 provides a variety of service functions related to assuring availability of the multimedia console 300. The audio processing unit 323 and an audio codec 332 form a corresponding audio process-
The front panel I/O subassembly 330 supports the functionality of the power button 353 and the eject button 352, as well as any LEDs (light emitting diodes) or other indicators exposed on the outer surface of the multimedia console 300. A system power supply module 336 provides power to the components of the multimedia console 300. A fan 338 cools the circuitry within the multimedia console 300.

The CPU 301, GPU 308, memory controller 310, and various other components within the multimedia console 300 are interconnected via one or more buses, including serial and parallel buses, a memory bus, a peripheral bus, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include a Peripheral Component Interconnect (PCI) bus, PCI-Express bus, etc.

When the multimedia console 300 is powered ON, application data can be loaded from the system memory 343 into memory 312 and/or caches 302, 304 and executed on the CPU 301. The application can present a graphical user interface that provides a consistent user experience when navigating to different media types available on the multimedia console 300. In operation, applications and/or other media contained within the media drive 344 can be launched or played from the media drive 344 to provide additional functionalities to the multimedia console 300.

The multimedia console 300 can be operated as a standalone system by simply connecting the system to a television or other display. In this standalone mode, the multimedia console 300 allows one or more users to interact with the system, watch movies, or listen to music. However, with the integration of broadband connectivity made available through the network interface 324 or the wireless adapter 348, the multimedia console 300 can further be operated as a participant in the larger network community, such as an online gaming community for example.

Fig. 4 and the following discussion provide a brief general description of a suitable computing environment in which in-play detection of altered game data can be implemented. Although not required, various aspects of in-play detection of altered game data can be described in the general context of computer executable instructions, such as program modules, being executed by a computer, such as a client workstation or a server. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, implementation of in-play detection of altered game data can be practiced with other computer system configurations, including hand held devices, multi processor systems, microprocessor based or programmable consumer electronics, network PCs, mini-computers, mainframe computers, and the like. Further, in-play detection of altered game data also can be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules can be located in both local and remote memory storage devices.

A computer system can be roughly divided into three component groups: the hardware component, the hardware/software interface system component, and the application programs component (also referred to as the “user component” or “software component”). In various embodiments of a computer system the hardware component may comprise the central processing unit (CPU) 521, the memory (both ROM 564 and RAM 525), the basic input/output system (BIOS 566), and various input/output (I/O) devices such as a keyboard 540, a mouse 542, a monitor 547, and/or a printer (not shown), among other things. The hardware component comprises the basic physical infrastructure for the computer system.

The applications programs component comprises various software programs including but not limited to compilers, database systems, word processors, business programs, videogames, and so forth. Application programs provide the means by which computer resources are utilized to solve problems, provide solutions, and process data for various users (machines, other computer systems, and/or end-users). In an example embodiment, application programs perform the functions associated with in-play detection of altered game data as described above.

The hardware/software interface system component comprises (and, in some embodiments, may solely consist of) an operating system that itself comprises, in most cases, a shell and a kernel. An “operating system” (OS) is a special program that acts as an intermediary between application programs and computer hardware. The hardware/software interface system component may also comprise a virtual machine manager (VMM), a Common Language Runtime (CLR) or its functional equivalent, a Java Virtual Machine (JVM) or its functional equivalent, or other such software components in the place of or in addition to the operating system in a computer system. A purpose of a hardware/software interface system is to provide an environment in which a user can execute application programs.

The hardware/software interface system is generally loaded into a computer system at startup and thereafter manages all of the application programs in the computer system. The application programs interact with the hardware/software interface system by requesting services via an application program interface (API). Some application programs enable end-users to interact with the hardware/software interface system via a user interface such as a command language or a graphical user interface (GUI).

A hardware/software interface system traditionally performs a variety of services for applications. In a multitasking hardware/software interface system where multiple programs may be running at the same time, the hardware/software interface system determines which applications should run in what order and how much time should be allowed for each application before switching to another application for a turn. The hardware/software interface system also manages the sharing of internal memory among multiple applications, and handles input and output to and from attached hardware devices such as hard disks, printers, and dial-up ports. The hardware/software interface system also sends messages to each application (and, in certain cases, to the end-user) regarding the status of operations and any errors that may have occurred. The hardware/software interface system can also offload the management of batch jobs (e.g., printing) so that the initiating application is freed from this work and can resume other processing and/or operations. On computers
that can provide parallel processing, a hardware/software interface system also manages dividing a program so that it runs on more than one processor at a time.

[0036] A hardware/software interface system shell (referred to as a “shell”) is an interactive end-user interface to a hardware/software interface system. (A shell may also be referred to as a “command interpreter” or, in an operating system, as an “operating system shell”.) A shell is the outer layer of a hardware/software interface system that is directly accessible by application programs and/or end-users. In contrast to a shell, a kernel is a hardware/software interface system’s innermost layer that interacts directly with the hardware components.

[0037] As shown in FIG. 4, an exemplary general purpose computing system includes a conventional computing device 560 or the like, including a processing unit 521, a system memory 562, and a system bus 523 that couples various system components including the system memory to the processing unit 521. The system bus 523 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 564 and random access memory (RAM) 525. A basic input/output system 566 (BIOS), containing basic routines that help to transfer information between elements within the computing device 560, such as during start up, is stored in ROM 564. The computing device 560 may further include a hard disk drive 527 for reading from and writing to a hard disk (hard disk not shown), a magnetic disk drive 528 (e.g., floppy drive) for reading from or writing to a removable magnetic disk 529 (e.g., floppy disk, removal storage), and an optical drive 530 for reading from or writing to a removable optical disk 531 such as a CD ROM or other optical media. The hard disk drive 527, magnetic disk drive 528, and optical disk drive 530 are connected to the system bus 523 by a hard disk drive interface 532, a magnetic disk drive interface 533, and an optical drive interface 534, respectively. The drives and their associated computer readable media provide non volatile storage of computer readable instructions, data structures, program modules and other data for the computing device 560. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 529, and a removable optical disk 531, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like may also be used in the exemplary operating environment. Likewise, the exemplary environment may also include many types of monitoring devices such as heat sensors and security or fire alarm systems, and other sources of information.

[0038] A number of program modules can be stored on the hard disk, magnetic disk 529, optical disk 531, ROM 564, or RAM 525, including an operating system 535, one or more application programs 536, other program modules 537, and program data 538. A user may enter commands and information into the computing device 560 through input devices such as a keyboard 540 and pointing device 542 (e.g., mouse). Other input devices (not shown) may include a microphone, joystick, game pad, satellite disk, scanner, or the like. These and other input devices are often connected to the processing unit 521 through a serial port interface 546 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port, or universal serial bus (USB). A monitor 547 or other type of display device is also connected to the system bus 523 via an interface, such as a video adapter 548. In addition to the monitor 547, computing devices typically include other peripheral output devices (not shown), such as speakers and printers. The exemplary environment of FIG. 4 also includes a host adapter 555, Small Computer System Interface (SCSI) bus 556, and an external storage device 562 connected to the SCSI bus 556.

[0039] The computing device 560 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 549. The remote computer 549 may be another computing device (e.g., personal computer), a server, a router, a network PC, a peer device, or other common network node, and typically includes many or all of the elements described above relative to the computing device 560, although only a memory storage device 550 (floppy drive) has been illustrated in FIG. 4. The logical connections depicted in FIG. 4 include a local area network (LAN) 551 and a wide area network (WAN) 552. Such networking environments are commonplace in offices, enterprise wide computer networks, intranets and the Internet.

[0040] When used in a LAN networking environment, the computing device 560 is connected to the LAN 551 through a network interface or adapter 553. When used in a WAN networking environment, the computing device 560 can include a modem 554 or other means for establishing communications over the wide area network 552, such as the Internet. The modem 554, which may be internal or external, is connected to the system bus 523 via the serial port interface 546. In a networked environment, program modules depicted relative to the computing device 560, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0041] While it is envisioned that numerous embodiments of in-play detection of altered game data are particularly well-suited for computerized systems, nothing in this document is intended to limit the invention to such embodiments. On the contrary, as used herein the term “computer system” is intended to encompass any and all devices capable of storing and processing information and/or capable of using the stored information to control the behavior or execution of the device itself, regardless of whether such devices are electronic, mechanical, logical, or virtual in nature.

[0042] The various techniques described herein can be implemented in connection with hardware or software or, where appropriate, with a combination of both. Thus, the methods and apparatuses for implementing in-play detection of altered game data, or certain aspects or portions thereof, can take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium, wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for implementing in-play detection of altered game data.

[0043] The program(s) can be implemented in assembly or machine language, if desired. In any case, the language can be a compiled or interpreted language, and combined with hardware implementations. The methods and apparatuses for
implementing in-play detection of altered game data also can be practiced via communications embodied in the form of program code that is transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via any other form of transmission, wherein, when the program code is received and loaded into and executed by a machine, such as an EPROM, a gate array, a programmable logic device (PLD), a client computer, or the like. When implemented on a general-purpose processor, the program code combines with the processor to provide a unique apparatus that operates to invoke the functionality of in-play detection of altered game data. Additionally, any storage techniques used in connection with in-play detection of altered game data can invariably be a combination of hardware and software.

While in-play detection of altered game data has been described in connection with the example embodiments of the various figures, it is to be understood that other similar embodiments can be used or modifications and additions can be made to the described embodiments for performing the same functions of in-play detection of altered game data without deviating therefrom. Therefore, in-play detection of altered game data as described herein should not be limited to any single embodiment, but rather should be construed in breadth and scope in accordance with the appended claims.

What is claimed:

1. A method for detecting altered data in memory of a game device, the method comprising:
   - locating the select data in memory;
   - generating an indication of the select data in memory;
   - providing the indication of the select data in memory;
   - if the indication of the select data in memory matches an expected indication of the select data in memory, continuing execution of the game; and
   - if the indication of the select data in memory does not match the expected indication of the select data in memory, halting execution of the game.

2. A method in accordance with claim 1, wherein execution of the game comprises online execution of the game.

3. A method in accordance with claim 1, wherein the challenge comprises an executable program configured to inspect the select data in memory.

4. A method in accordance with claim 1, wherein:
   - the indication of the select data in memory comprises at least one cryptographic hash value indicative of the select data in memory;
   - the expected indication of the select data in memory comprises at least one cryptographic hash value indicative of the expected select data in memory; and
   - comparing the indication of the select data in memory with the expected indication of the select data in memory comprises respectively comparing at least one of the at least one hash value indicative of the select data in memory with at least one of the at least one expected hash value.

5. A method in accordance with claim 1, wherein the select data in memory is located via an obfuscated lookup table.

6. A method in accordance with claim 1, wherein the select data comprises at least one of a game constant and a game characteristic.

7. A method in accordance with claim 1, wherein the challenge comprises an indication of at least one name of a portion of the select data.

8. A device for detecting altered game data, the system comprising:
   - a processing portion configured to:
     - locate select data in a memory of the device;
     - execute an executable program received via a challenge to inspect the select data in the memory of the device;
     - generate an indication of the select data in the memory in accordance with the executable program;
     - an input/output portion configured to:
       - during execution of a game on the device, receive the challenge;
       - provide an indication of the select data in the memory determined in accordance with the executable program;
       - receive an indication that a comparison of the indication of the select data in the memory with an expected indication of select data in the memory do match;
       - receive an indication that a comparison of the indication of the select data in the memory with an expected indication of select data in the memory do not match; and
       - receive an indication that a comparison of an indication of the select data in the memory portion with an expected indication of select data in memory do match; and
   - the memory configured to store the select data.

9. A device in accordance with claim 8, wherein execution of the game comprises online execution of the game.

10. A device in accordance with claim 9, wherein:
    - if the indication of the select data in memory matches the expected indication of the select data, online gaming is allowed to continue; and
    - if the indication of the select data in memory does not match the expected indication of the select data, online gaming is not allowed to continue.

11. A device in accordance with claim 8, wherein:
    - the indication of the select data in memory comprises at least one cryptographic hash value indicative of the select data in memory;
    - the expected indication of the select data in the memory comprises at least one cryptographic hash value indicative of the expected select data in memory; and
    - comparing the indication of the select data in memory with the expected indication of the select data in memory comprises respectively comparing at least one of the at least one hash value indicative of the select data in memory with at least one of the at least one hash value indicative of the expected select data in memory.

12. A device in accordance with claim 8, wherein the select data in memory is located via an obfuscated lookup table.

13. A device in accordance with claim 8, wherein the select data comprises at least one of a game constant and a game characteristic.

14. A device in accordance with claim 8, wherein the challenge comprises an indication of at least one name of a portion of the select data.

15. A computer-readable medium having stored thereon computer-executable instruction for detecting altered data by performing the steps of:
    - during execution of an online game, receiving a challenge to inspect select data of in-memory data;
    - locating the select data in memory;
    - generating an indication of the select data in memory;
    - providing the indication of the select data in memory;
if the indication of the select data in memory matches an expected indication of the select data in memory, continuing execution of the game; and
if the indication of the select data in memory does not match the expected indication of the select data in memory, halting execution of the game.

16. A computer-readable medium in accordance with claim 15, wherein the challenge comprises an executable program configured to inspect the select data.

17. A computer-readable medium in accordance with claim 15, wherein:
the indication of the select data in memory comprises at least one cryptographic hash value indicative of the select data in memory;
the expected indication of the select data in memory comprises at least one cryptographic hash value indicative of the expected select data in memory; and
comparing the indication of the select data in memory with the expected indication of the select data comprises respectively comparing at least one of the at least one hash value indicative of the select data in memory with at least one of the at least one expected hash value.

18. A computer-readable medium in accordance with claim 15, the computer-executable instructions further for locating the select data in memory via an obfuscated lookup table.

19. A computer-readable medium in accordance with claim 15, wherein the select data comprises at least one of a game constant and a game characteristic.

20. A computer-readable medium in accordance with claim 15, wherein the challenge comprises an indication of at least one name of a portion of the select data.

* * * * *