

Block Chain-based File Protection Method and Terminal Device

The application claims priority for Chinese Patent Application submitted to the National Intellectual Property Administration, PRC on May 22, 2018, Application No. 201810496143.4, and
5 titled “Block Chain-based File Protection Method and Terminal Device”, with all its contents hereby incorporated by reference.

TECHNICAL FIELD

The application belongs to the technical field of computers, and particularly relates to a block chain-based file protection method and a terminal device.

10

BACKGROUND

With the development of mobile Internet, applications (Apps) for providing various services have emerged. Users can download and install the APPs on terminals to enjoy corresponding services provided by these APPs, such as block chain Apps, taxi-hailing Apps, and financial Apps.

However, hackers or outlaws may attack the Apps installed on the terminals with de-compilation
15 tools to acquire an executable file in an installation package of the Apps, and then de-compile the executable file to obtain a source code of the executable file to steal an algorithm in the source code or to tamper the source code to steal user information or user properties in the Apps, which causes severe economic losses to users and leads to low data security.

The application provides a block chain-based file protection method and a terminal device to
20 solve the problem of low data security caused by source code leakage resulting from de-compilation of executable files of Apps in the prior art.

SUMMARY

In the first aspect, the application provides a block chain-based file protection method comprising:

Acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for
5 identifying a programming language adopted by the source file corresponding to the first file;

Determining protection information of the first file according to the type information, wherein the protection information is used for preventing de-compilation; and

Configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the
10 first file.

In the second aspect, the application provides a terminal device comprising:

An acquisition unit used for acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be
15 executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

A protection information determination unit used for determining protection information of the first file according to the type information, wherein the protection information is used for preventing
20 de-compilation; and

A configuration unit used for configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

In the third aspect, the application provides a terminal device comprising a memory, a processor,
25 and a computer-readable instruction which is stored in the memory and is able to run in the processor,

wherein the processor executes the computer-readable instruction to implement the following steps:

Acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

Determining protection information of the first file according to the type information, wherein the protection information is used for preventing de-compilation; and

Configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

In the fourth aspect, the application provides a computer-readable storage medium having a computer-readable instruction stored therein, wherein the computer-readable instruction is executed by a processor to implement the following steps:

Acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

Determining protection information of the first file according to the type information, wherein the protection information is used for preventing de-compilation; and

Configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

According to the application, protection information is configured into the first file used for

establishing the data interaction port of the block chain to alias contents in the first file, so that an algorithm in the first file will not be leaked when a source code is exposed by de-compilation. As a binary file, the first file, instead of the source code of the first file, is processed to improve the complexity and difficulty of acquiring the source code by de-compiling the first file, and thus, the data security of the first file is improved. The first interaction file is used for establishing the data interaction port of the block chain, the security of data transmitted via the data interaction port of the block chain can be improved while the security of the first file is improved, and accordingly, the security of user information or user properties in an App corresponding to the first file is improved.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 is an implementation flow diagram of a block chain-based file protection method in one embodiment of the application;

FIG. 2 is a specific implementation flow diagram of S103 of the block chain-based file protection method in the embodiment of the application;

FIG. 3 is an implementation flow diagram of the block chain-based file protection method in another embodiment of the application;

FIG. 4 is a schematic block diagram of a terminal device in one embodiment of the application;

FIG. 5 is a schematic block diagram of the terminal device in another embodiment of the application.

DESCRIPTION OF EMBODIMENTS

For the sake of a thorough understanding of the embodiments of the application, specific details such as specific system structures and techniques are given in the following description for the purpose of explanation instead of limitation. Those skilled in the art would appreciate that the application can also be implemented in other embodiments without these specific details. A detailed description of the common systems, devices, circuits and methods in other cases is omitted to avoid influences of unnecessary details on the description of the application.

Please refer to FIG. 1 which is an implementation flow diagram of a block chain-based file protection method in one embodiment of the application. The block chain-based file protection method in this embodiment is executed by a terminal device which is a mobile terminal including, but not limited to, a smart phone, a tablet computer, and a notebook computer, or a desktop computer. As shown in FIG. 1, the block chain-based file protection method comprises the following steps:

S101: type information of a first file to be protected is acquired, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file.

When detecting that a user triggers a file strengthening instruction through an interface, the terminal device acquires a to-be-protected first file selected by the user and acquires type information of the first file. The user is a developer, a tester, or a software manager. The file strengthening instruction is used for strengthening the to-be-protected file to prevent the to-be-protected file from being de-compiled, which may otherwise expose a source program.

Wherein, the block chain simply includes, by architecture, a protocol layer, an extension layer and an application layer, and the first file is used for establishing a data interaction port between the extension layer and the application layer, so that the source code is prevented from being exposed after the first file is de-compiled, which may otherwise lead to leakage of data uploaded by the user or data stored in the extension layer.

The programming language adopted by the source file corresponding to the first file is an assembly language or a high-level language, such as Java language or GO language.

Particularly, in one embodiment, a file header of the first file includes descriptive information used for indentifying the programming language adopted by the source file corresponding to the first file, and the terminal device reads the descriptive information from the file header of the first file and determines the programming language adopted by the source file corresponding to the first file

according to the descriptive information.

In another embodiment, the terminal device scans the contents in the first file to extract characteristic information from the contents and determines the type information of the first file according to the characteristic information. The characteristic information is a key field which identifies the type or version number of a compile.

S102: protection information of the first file is determined according to the type information, wherein the protection information is used for preventing de-compilation.

When receiving the type information of the first file, the terminal device selects protection information matching the type information from a database and determines protection information of the first file from the matching protection information. In the case where at least two pieces of matching protection information are acquired, one of the at least two pieces of matching protection information is randomly selected by the terminal device to serve as the protection information of the first file; or, the at least two pieces of matching protection information are flagged to be sequentially and cyclically used, for example, a first piece of matching protection information is used as the protection information corresponding to the type information when the terminal device performs S102 for the first time, a second piece of matching protection information is used as the protection information corresponding to the type information when the terminal device performs S102 for the second time, and so on.

Understandably, the protection information includes an invalid program fragment or data obtained after the invalid program fragment is compiled. The invalid program fragment is used for aliasing or modifying an execution path of the contents in the first file to increase the difficulty of analyzing binary data corresponding to the first file, so that the source code corresponding to the first file is protected against exposure when the first file is de-compiled; and the invalid program fragment will not change the execution logic of the contents in the first file, so that the algorithm implementation logic of the first file is maintained. The invalid program fragment is a meaningless program instruction or an invalid program instruction having an execution logic. There is one or at

least two program fragments. For example, in the case where the programming language adopted by the source file corresponding to the first file is GO language, a program fragment compiled in GO language is selected from the database. In the case where the programming language adopted by the source file corresponding to the first file is Java language, a program fragment compiled in Java language is selected from the database.

In another embodiment, S102 is specifically implemented as follows: interference information to be inserted into the first file is randomly generated according to the type information, wherein the interference information is used for aliasing the execution path of the contents in the first file.

For example, the terminal device randomly generates protection information matching the type information according to the type information of the first file. The randomly-generated protection information includes interference information which is an invalid program fragment, and data lengths of generated invalid program fragments are identical or different. The contents of the randomly-generated interference information are different. For example, a program fragment in the protection formation, matching the type information of the first file, randomly generated when the terminal device performs S102 for the first time has a first length, and a program fragment in the protection formation, matching the type information of the first file, randomly generated when the terminal device performs S102 for the second time has a second length.

S103: the protection information is configured into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

A preset configuration strategy is pre-stored in the terminal device and is used for indicating how to configure the determined protection information. The preset configuration strategy can be set as actually required, and the application has no limitation in this aspect.

Understandably, the terminal device can mark, in the second file, the position or address of the protection information configured into the first file, so that the configured protection information can be accurately recognized when the terminal device loads the second file or an executable file

corresponding to the second file; and the execution logic of the second file is the same as that of the first file to keep the execution logic of an algorithm or a principal function unchanged before and after the protection information is added, so that a preset operation of the algorithm or the principal function can be performed to realize an expected function.

5 In another embodiment, in order to improve the data security of the first file to increase the difficulty of analyzing the first file after de-compilation, the protection information includes an invalid program fragment used for aliasing the execution path of the contents in the first file without changing the execution logic of the first file. S103 may comprise S1031-S1032. Particularly, referring to FIG. 2 which is specific implementation flow diagram of S103 of the block chain-based
10 file protection method in the embodiment of the application, S103 particularly comprises:

S1031: location information of the invalid program fragment to be inserted is determined.

The preset configuration strategy is used for indicating that the invalid program fragment is to be configured into the head or tail of a data area of the first file, and in this case, the location information of the invalid program fragment to be inserted corresponds to the head or tail of the data area of the
15 first file; or, the preset configuration strategy is used for indicating that the invalid program fragment is to be configured into binary data corresponding to a to-be-protected function in the first file, and in this case, the location information of the invalid program fragment to be inserted corresponds to the corresponding position of the to-be-protected function in the first file; or, the preset configuration strategy is used for indicating that binary information of the first file is to be grouped and the invalid
20 program fragment is to be inserted between any two data groups, and in this case, the location information of the invalid program fragment to be inserted corresponds to the junction of the two data groups having the invalid program fragments to be inserted therebetween.

S1032: the invalid program fragment is configured into the first file according to the location information to obtain a second file.

25 For example, in the case where the terminal device determines that the preset configuration strategy indicates that the protection information is to be configured into the head or tail of the data

area of the first file, if the protection information of the first file includes an invalid program fragment, the invalid program fragment is compiled through a compiling tool to obtain interference data which is inserted into the head or tail of the data area of the first file; or, if the protection information of the first file includes the interference data obtained after the invalid program fragment is compiled, the interference data is directly inserted into the head or tail of the data area of the first file. After the interference data is inserted into the head or tail of the data area of the first file, the processed first file is compiled by the terminal device through a compiler to obtain the second file.

In the case where the terminal device determines that the preset configuration strategy indicates that the protection information is to be configured into the binary data corresponding to the to-be-protected function in the first file, if the protection information of the first file includes an invalid program fragment, the invalid program fragment is compiled through a compiling tool to obtain interference data which is inserted into the binary data corresponding to the to-be-protected function of the first file; or, if the protection information of the first file includes the interference data obtained after the invalid program fragment is compiled, the interference data is directly inserted into the binary data corresponding to the to-be-protected function of the first file. After the interference data is inserted into the binary data corresponding to the to-be-protected function of the first file, the processed first file is compiled by the terminal device through a compiler to obtain the second file.

In the case where the terminal device determines that preset configuration strategy indicates that the binary data of the first file is to be grouped and the protection information is to be configured into between any two data groups, the terminal device divides the binary data in the data area of the first file into a preset number of data groups, and if the protection information of the first file includes an invalid program fragment, the invalid program fragment is compiled through a compiling tool to obtain interference data which is inserted between any two corresponding data groups; if the protection information of the first file includes the interference data obtained after the invalid program fragment is compiled, the interference data is directly inserted between any two corresponding data groups of the first file. After the interference data is inserted between any two corresponding data groups of the first file, the processed first file is compiled by the terminal device

through a compiler to obtain the second file.

Furthermore, when the protection information includes the invalid program fragment or the interference data obtained after the invalid program fragment is compiled, information used for indicating an execution strategy of the protection information is set when the terminal device
5 configures the protection device, and S1032 particularly comprises the following steps:

Flag information of the invalid program fragment is determined according to the invalid program fragment and the location information, wherein the flag information includes an identifier or a flag value and is used for indicating the execution strategy of the invalid program fragment ;

The invalid program fragment and the flag information of the invalid program fragment are
10 configured into the first file according to the location information to obtain the second file.

For example, when the terminal device configures the protection information, a position or address corresponding to the protection information is flagged, a flag bit used for indicating the execution strategy of the protection information is set according to the type of the protection information, and an identifier or a flag value is set at the flag bit. The position or address
15 corresponding to the protection information is flagged for the purpose of accurately recognizing the protection information, and the execution strategy of the protection information indicates that whether or not the protection information is to be executed or how to execute the protection information. For example, when the flag value of the flag bit is 0, it indicates that the protection information is not to be executed; or, when the flag value of the flag bit is 1, it indicates that the
20 protection information is to be executed.

In one embodiment, assume the data area of the first file includes a binary data group a and a binary data group b, in the case where the protection information is the interference data obtained after the invalid program fragment is compiled and the invalid program fragment has an execution logic, the terminal device can, if the interference data c obtained by the compilation of the invalid
25 program fragment is inserted between the binary data group a and the binary group b of the first file, set a first identifier at the tail of the binary data group a and a second identifier at the tail of the

interference data c obtained by the compilation of the invalid program fragment.

Wherein, program instructions corresponding to the binary data group a and the binary data group b are complete. The first identifier is used for indicating a jump, after the instruction corresponding to the binary data group a is executed, to an entry address corresponding to the invalid program fragment to execute the invalid program fragment. The second identifier is used for indicating a jump, after the invalid program fragment is executed, to the program instruction corresponding to the binary data group b to execute the program instruction corresponding to the binary data group b.

Before the insertion of the interference data c obtained by the compilation of the invalid program fragment, the execution path of the first file is as follows: binary data group a, and binary data group b. After the insertion of the interference data c obtained by the compilation of the invalid program fragment, the execution path of the second file is correspondingly changed into: binary data group a, interference data c, and binary data group b.

In another embodiment, in the case where the protection information is the interference data obtained after the invalid program fragment is compiled and the invalid program fragment does not have an execution logic, the invalid program fragment is an unreadable code or a junk code, the terminal device does not need to execute the invalid program fragment, and therefore, the terminal device sets an identifier or flag value at the head of the invalid program fragment to indicate that the invalid program fragment is not to be executed.

When the first file is a target file and the second file obtained by configuration of the protection information is also a target file, the terminal device sets, in a symbol area of the second file, an address indicating the location of the protection information and an identifier indicating the execution strategy of the protection information and sets, in a symbol descriptor table, the identifier indicating the execution strategy of the protection information in the connecting process of the second file to form an executable file, so that when the executable file corresponding to the second file is loaded, the terminal device can recognize the protection information according to the address of the position

where the protection information is located, can query the identifier indicating the execution strategy of the protection information from the symbol descriptor table of the executable file corresponding to the second file, and can analyze the identifier to obtain the execution strategy of the protection information so as to execute the protection information according to the execution strategy of the protection information.

When the first file is an executable file, the terminal device sets, in the symbol descriptor table of the second file, relevant information of the identifier indicating the execution strategy of the protection information. For example, the relevant information may include an address field corresponding to the protection information and an address corresponding to the flag bit, and may also include an identifier corresponding to the flag bit. The address field corresponding to the protection information is used for recognizing the protection information, and the address corresponding to the flag bit is used for acquiring the flag value or the identifier, so that when the flag value or the execution strategy, identified by the identifier, of the protection information is obtained by analysis, processing can be performed according to the execution strategy of the protection information obtained by analysis.

According to the above solution, protection information is configured in the first file used for establishing the data interaction port of the block chain to alias contents in the first file, so that an algorithm in the first file will not be leaked when a source code is exposed by de-compilation. As a binary file, the first file, instead of the source code of the first file, is processed to improve the complexity and difficulty of acquiring the source code by de-compiling the first file, and thus, the data security of the first file is improved. The first interaction file is used for establishing the data interaction port of the block chain, the security of data transmitted via the data interaction port of the block chain can be improved while the security of the first file is improved, and accordingly, the security of user information or user properties in an App corresponding to the first file is improved.

Referring to FIG. 3 which is an implementation flow diagram of the block chain-based file protection method in another embodiment of the application, this embodiment is different from the above embodiment in the following aspects: before the protection information is configured into the

first file according to the preset configuration strategy to obtain the second file, an entry address of a target function corresponding to to-be-protected target data is hidden or modified. Particularly, the block chain-based file protection method shown in FIG. 3 specifically comprises the following steps:

5 S201: type information of a first file to be protected is acquired, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

10 S201 in this embodiment is the same as S101 in the above embodiment, can be implemented with reference to relevant description of S101 in the above embodiment, and will not be described anymore herein.

S202: protection information of the first file is determined according to the type information, wherein the type information is used for preventing de-compilation.

15 S202 in this embodiment is the same as S102 in the above embodiment, can be implemented with reference to relevant description of S102 in the above embodiment, and will not be described anymore herein..

S203: to-be-protected target data in the first file is determined, wherein the target data is used for realizing a to-be-protected target function.

20 The target function is a principal function or a function including a core algorithm. The terminal device determines the to-be-protected target data in the first file according to an identifier of the target function or determines the to-be-protected target data in the first file according to an address field corresponding to the target function in the first file.

25 Understandably, the sequence of performing S201 and S203 is not limited. That is, S201-S202 are performed first, and then S203 is performed; or, S203 is performed first, and then S201-S202 are performed; or, S201 and S203 are performed synchronously. The application has no limitation in this

aspect.

S204, the entry address of the target function is hidden or modified, and identification information is added into the first file, wherein the identification information is used for describing the entry address.

5 Particularly, when the terminal device determines the to-be-protected target data in the first file according to the identifier of the target function, a target address field distributed to the target data is acquired, an address offset of the target data is set, and then the target address field is modified according to the set address offset, so that the entry address of the target function is modified. The address offset is randomly distributed or is a preset value, and the application has no limitation in this
10 aspect.

When the terminal device determines the to-be-protected target data in the first file according to the address field corresponding to the target function in the first file, an address offset of the target data is set, and then the target address field is modified according to the set address offset, so that the entry address of the target function is modified.

15 After the entry address of the target function is modified, identification information used for describing the address offset of the entry address is added into a symbol area of the first file according to the address offset.

The terminal device can also insert invalid data in front of the target address field according to the target address field distributed to the target data to hide the entry address of the target function
20 corresponding to the target data.

S205: the protection information is configured into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

S205 in this embodiment is the same as S103 in the above embodiment, can be implemented
25 with reference to relevant description of S103 in the above embodiment, and will not be described

anymore herein..

According to the above solution, protection information is configured in the first file used for establishing the data interaction port of the block chain to alias contents in the first file, so that an algorithm in the first file will not be leaked when a source code is exposed by de-compilation. As a binary file, the first file, instead of the source code of the first file, is processed to improve the complexity and difficulty of acquiring the source code by de-compiling the first file, and thus, the data security of the first file is improved. The first interaction file is used for establishing the data interaction port of the block chain, the security of data transmitted via the data interaction port of the block chain can be improved while the security of the first file is improved, and accordingly, the security of user information or user properties in an App corresponding to the first file is improved.

The terminal device hides or modifies the entry address of the target function corresponding to the to-be-protected target data, so that the difficulty of acquiring and analyzing the target data is increased, and the security of the target data is improved.

Understandably, the serial numbers of the steps in the above embodiments do not indicate the executing sequence of these steps. The executing sequence of the process is determined by function and internal logic, and is not intended to limit the implementation process of the embodiments of the application in any way.

As shown in FIG. 4 which is a schematic block diagram of a terminal device in one embodiment of the application, all units included in the terminal device are used for performing the steps of the block chain-based file protection method in any above-mentioned embodiment of the application. Please refer to relevant description in the embodiments corresponding to FIG. 1 to FIG. 3 for specific details. For the sake of a brief description, only parts relating to this embodiment are shown. The terminal device 4 in this embodiment comprises an acquisition unit 410, a protection information determination unit 420 and a configuration unit 430, wherein:

The acquisition unit 410 is used for acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file

or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

5 The protection information determination unit 420 is used for determining protection information of the first file according to the type information, wherein the protection information is used for preventing de-compilation;

10 Furthermore, the protection information determination unit 420 is specifically used for randomly generating interference information to be inserted into the first file according to the type information, wherein the interference information is used for aliasing an execution path of contents in the first file; and

 The configuration unit 430 used for configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

15 Furthermore, the protection information includes an invalid program fragment, and the configuration unit 430 comprises:

 A location information determination unit 431 used for determining location information of the invalid program fragment to be inserted; and

20 An information configuration unit 432 used for configuring the invalid program fragment into the first file according to the location information to obtain a second file.

 Furthermore, the information configuration unit 432 is particularly used for:

 Determining flag information of the invalid program fragment according to the invalid program fragment and the location information, wherein the flag information includes an identifier or a flag value and is used for indicating an execution strategy of the invalid program fragment; and

25 Configuring the invalid program fragment and the flag information of the invalid program

fragment into the first file according to the location information to obtain a second file.

Optionally, the terminal device further comprises:

A target data determination unit 440 used for determining to-be-protected target data in the first file, wherein the target data is used for realizing a to-be-protected target function; and

5 A setting unit 450 used for hiding or modifying an entry address of the target function and adding identification information into the first file, wherein the identification information is used for describing and acquiring the entry address.

According to the above solution, protection information is configured in the first file used for establishing the data interaction port of the block chain to alias contents in the first file, so that an
10 algorithm in the first file will not be leaked when a source code is exposed by de-compilation. As a binary file, the first file, instead of the source code of the first file, is processed to improve the complexity and difficulty of acquiring the source code by de-compiling the first file, and thus, the data security of the first file is improved. The first interaction file is used for establishing the data
15 interaction port of the block chain, the security of data transmitted via the data interaction port of the block chain can be improved while the security of the first file is improved, and accordingly, the security of user information or user properties in an App corresponding to the first file is improved.

The terminal device hides or modifies the entry address of the target function corresponding to the to-be-protected target data, so that the difficulty of acquiring and analyzing the target data is increased, and the security of the target data is improved.

20 FIG. 5 is a schematic block diagram of the terminal device in another embodiment of the application. As shown in FIG. 5, the terminal device 5 in this embodiment comprises a processor 510, a memory 520, and a computer-readable instruction 530 which is stored in the memory 520 and is able to run in the processor 510, such as a program of the block chain-based file protection method. The processor 510 executes the computer-readable instruction 530 to implements the steps of the
25 embodiments of the block chain-based file protection method, such as S101 to S103 shown in FIG. 1. Or, the processor 510 executes the computer-readable instruction 530 to fulfill the functions of the

units shown in FIG. 4, such as the functions of units 410-450 shown in FIG. 4. Please refer to relevant description in the embodiment shown in FIG. 4 for specific details, and repeated description will not be given anymore herein.

Illustratively, the computer-readable instruction 530 can be divided into one or more units which
5 are stored in the memory 520 and are to be executed by the processor 510 to implement the application. The one or more units can be a series of instruction fields, capable of fulfilling specific functions, of the computer-readable instruction, and the instruction fields are used for describing the execution process of the computer-readable instruction 530 in the terminal device 5. For example, the computer-readable instruction 530 is divided into an acquisition unit, a protection information
10 determination unit and a configuration unit, and the units have the functions as described above.

The terminal device includes, but is not limited to, a processor 510 and a memory 520. Those skilled in the art would appreciate that FIG. 5 is only an illustration of the terminal device 5, and is not intended to limit the terminal device 5. The terminal device 5 may include more or less components against FIG. 5, or some components may be combined, or the components may be configured in
15 different manners. For example, an air conditioner further includes an input-output device, a network access device, a bus, and the like.

The processor 510 is a central processing unit (CPU) or other universal processors, a digital signal processor (DSP), an application special integrated circuit (ASIC), a field-programmable gate array (FPGA) or other programmable logic devices, a discrete gate or transistor logic device, or a
20 discrete hardware component. The universal processors are microprocessors or any conventional processors.

The memory 520 is an internal memory cell of the terminal device 5, such as a hard disk or internal memory of the terminal device 5. Or, the memory 520 is an external memory device of the terminal device 5, such as a plug-in hard disk, a smart media card (SMC), a secure digital (SD) card,
25 or a flash card installed on the terminal device 5. Furthermore, the memory 520 includes an internal memory cell of the terminal device and an external memory device. The memory 520 is used for

storing the computer-readable instruction and other programs and data required by the air conditioner and is also used for temporarily storing output data or to-be-output data.

The above embodiments are only used for explaining, instead of limiting, the technical solutions of the application. Although the application is detailed with reference to the above embodiments, 5 those ordinarily skilled in the art would appreciate that the technical solutions in the above embodiments can be modified, or part of technical characteristics can be equivalently replaced; and all these modifications or replacements achieved without deviating from the spirit and scope of the technical solutions of the application should also fall within the protection scope of the application.

CLAIMS

1. A block chain-based file protection method, comprising:

acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target
5 file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

determining protection information of the first file according to the type information, wherein the protection information is used for preventing de-compilation; and

10 configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

2. The block chain-based file protection method according to Claim 1, wherein the step of determining protection information of the first file according to the type information comprises:

15 randomly generating interference information to be inserted into the first file according to the type information, wherein the interference information is used for aliasing an execution path of contents in the first file.

3. The block chain-based file protection method according to Claim 1 or 2, wherein the protection information includes an invalid program fragment, and the step of configuring the
20 protection information into the first file according to the preset configuration strategy to obtain the second file comprises:

determining location information of the invalid program fragment to be inserted; and

configuring the invalid program fragment into the first file according to the location information to obtain the second file.

4. The block chain-based file protection method according to Claim 3, wherein the step of configuring the protection information into the first file according to the preset configuration strategy to obtain the second file comprises:

determining flag information of the invalid program fragment according to the invalid program
5 fragment and the location information, wherein the flag information includes an identifier or a flag value and is used for indicating an execution strategy of the invalid program fragment; and

configuring the invalid program fragment and the flag information of the invalid program fragment into the first file according to the location information to obtain the second file.

5. The block chain-based file protection method according to Claim 1, wherein the method
10 further comprises the following steps performed before the step of configuring the protection information into the first file according to the preset configuration strategy to obtain the second file:

determining to-be-protected target data in the first file, wherein the target data is used for realizing a to-be-protected target function; and

hiding or modifying an entry address of the target function, and adding identification
15 information into the first file, wherein the identification information is used for describing and acquiring the entry address.

6. A terminal device, comprising:

an acquisition unit used for acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a
20 executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

a protection information determination unit used for determining protection information of the
25 first file according to the type information, wherein the protection information is used for preventing

de-compilation; and

a configuration unit used for configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

5 7. The terminal device according to Claim 6, wherein the protection information determination unit is specifically used for:

randomly generating interference information to be inserted into the first file according to the type information, wherein the interference information is used for aliasing an execution path of contents in the first file.

10 8. The terminal device according to Claim 6 or 7, wherein the protection information includes an invalid program fragment, and the configuration unit comprises:

a location information determination unit used for determining location information of the invalid program fragment to be inserted; and

15 an information configuration unit used for configuring the invalid program fragment into the first file according to the location information to obtain the second file.

9. The terminal device according to Claim 8, wherein the information configuration unit is specifically used for:

20 determining flag information of the invalid program fragment according to the invalid program fragment and the location information, wherein the flag information includes an identifier or a flag value and is used for indicating an execution strategy of the invalid program fragment; and

configuring the invalid program fragment and the flag information of the invalid program fragment into the first file according to the location information to obtain the second file.

10. The terminal device according to Claim 6, wherein the block chain-based file protection device further comprises:

a target data determination unit used for determining to-be-protected target data in the first file, wherein the target data is used for realizing a to-be-protected target function; and

a setting unit used for hiding or modifying an entry address of the target function and adding identification information into the first file, wherein the identification information is used for
5 describing and acquiring the entry address.

11. A terminal device, comprising a memory, a processor, and a computer-readable instruction which is stored in the memory and is able to run in the processor, wherein the processor executes the computer-readable instruction to implement the following steps:

acquiring type information of a first file to be protected, wherein the first file is used for
10 establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for identifying a programming language adopted by the source file corresponding to the first file;

determining protection information of the first file according to the type information, wherein
15 the protection information is used for preventing de-compilation; and

configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the first file.

12. The terminal device according to Claim 11, wherein the step of determining the protection
20 information of the first file according to the type information comprises:

randomly generating interference information to be inserted into the first file according to the type information, wherein the interference information is used for aliasing an execution path of contents in the first file.

13. The terminal device according to Claim 11 or 12, wherein the protection information
25 includes an invalid program fragment, and the step of configuring the protection information into the

first file according to the preset configuration strategy to obtain the second file comprises:

determining location information of the invalid program fragment to be inserted; and

configuring the invalid program fragment into the first file according to the location information to obtain the second file.

5 14. The terminal device according to Claim 13, wherein the step of configuring the protection information into the first file according to the preset configuration strategy to obtain the second file comprises:

10 determining flag information of the invalid program fragment according to the invalid program fragment and the location information, wherein the flag information includes an identifier or a flag value and is used for indicating an execution strategy of the invalid program fragment; and

configuring the invalid program fragment and the flag information of the invalid program fragment into the first file according to the location information to obtain the second file.

15 15. The terminal device according to Claim 11, wherein before the step of configuring the protection information into the first file according to the preset configuration strategy to obtain the second file, the processor executes the computer-readable instruction to implement the following steps:

determining to-be-protected target data in the first file, wherein the target data is used for realizing a to-be-protected target function; and

20 hiding or modifying an entry address of the target function, and adding identification information into the first file, wherein the identification information is used for describing and acquiring the entry address.

16. A computer-readable storage medium, having a computer-readable instruction stored therein, wherein the computer-readable instruction is executed by at least one processor to implement the following steps:

acquiring type information of a first file to be protected, wherein the first file is used for establishing a data interaction port of a block chain and is a target file or a executable file, the target file is a binary file which is generated by compiling a source file and is to be executed by a processor, the executable file is formed after linking of the target file, and the type information is used for
5 identifying a programming language adopted by the source file corresponding to the first file;

determining protection information of the first file according to the type information, wherein the protection information is used for preventing de-compilation; and

configuring the protection information into the first file according to a preset configuration strategy to obtain a second file, wherein an execution logic of the second file is the same as that of the
10 first file.

17. The computer-readable storage medium according to Claim 16, wherein the step of determining the protection information of the first file according to the type information comprises:

randomly generating interference information to be inserted into the first file according to the type information, wherein the interference information is used for aliasing an execution path of
15 contents in the first file.

18. The computer-readable storage medium according to Claim 16 or 17, wherein the protection information includes an invalid program fragment, and the step of configuring the protection information into the first file according to the preset configuration strategy to obtain the second file comprises:

20 determining location information of the invalid program fragment to be inserted; and

configuring the invalid program fragment into the first file according to the location information to obtain the second file.

19. The computer-readable storage medium according to Claim 18, wherein the step of configuring the protection information into the first file according to the preset configuration strategy
25 to obtain the second file comprises:

determining flag information of the invalid program fragment according to the invalid program fragment and the location information, wherein the flag information includes an identifier or a flag value and is used for indicating an execution strategy of the invalid program fragment; and

5 configuring the invalid program fragment and the flag information of the invalid program fragment into the first file according to the location information to obtain a second file.

20. The computer-readable storage medium according to Claim 17, wherein before the step of configuring the protection information into the first file according to the preset configuration strategy to obtain the second file, the processor executes the computer-readable instruction to implement the following steps:

10 determining to-be-protected target data in the first file, wherein the target data is used for realizing a to-be-protected target function; and

hiding or modifying an entry address of the target function, and adding identification information into the first file, wherein the identification information is used for describing and acquiring the entry address.

15