

PROCESSING METHOD FOR PREVENTING COPY ATTACK, AND SERVER AND CLIENT

TECHNICAL FIELD

[0001] The present application relates to the field of security authentication
5 information data processing technologies, and in particular, to a processing method
for preventing a copy attack, a server, and a client.

BACKGROUND

[0002] The rapid development of the Internet and information technologies
increases dependence on mobile devices such as a mobile phone and a tablet in daily
10 life. An application (APP) related to user property is usually installed on a mobile
device of the user, for example, a mobile wallet, a mobile bank, an investment
application, or a financial management application. How to protect a user terminal
device from a hacker attack, avoid a user property loss, and effectively improve user
property security has become an important subject of current terminal information
15 security.

[0003] Currently, the APP of the mobile device usually uses soft secure storage
for secure data storage. The soft security storage generally means that an anti-cracking
method is added to software running on a device system such as an APP running on
ANDROID, to provide security for local secure data storage. Usually, it is difficult for
20 hacker to crack data and algorithms stored in the software. However, currently, hacker
can easily figure out a copy attack method. To be specific, a hacker copies data and a
user environment, and can directly simulate user equipment. As a result, the hacker
can easily counterfeit the user by using the user data in the device to steal user
information. The hacker can directly run a user account on a machine through copy
25 attack, instead of forcefully cracking the data and algorithm in the software. Existing
methods for preventing a copy attack are generally to bind with user equipment
information. If it is determined that a device is not the previously bound device, the
device is considered to be insecure. The existing methods can be easily cracked by

hackers because device environment information is obtained by using an API of the system. A hacker can invoke the API of the system by using a more accurate algorithm through a hook to obtain the device environment information, and simulate the device information by using the API, to make the software misrecognize the device as the previously bound device, thereby cracking device binding protection and entering the user account.

[0004] Therefore, in the existing technology, simply perform soft secure storage of data on a client is still relatively insecure, and it is difficult to prevent a copy attack. In addition, device information bound to a user is usually inherent; therefore, once an attacker (such as a hacker) successfully launches a copy attack, the user can suffer from an inestimable loss.

SUMMARY

[0005] The present application aims to provide a processing method for preventing a copy attack, a server, and a client, to increase difficulty in launching a copy attack, reduce harm of implementing a copy attack, improve security of data storage applied in a terminal device, reduce a user property loss, and protect user property security.

[0006] The processing method for preventing a copy attack, the server, and the client provided in the present application are implemented as follows:

[0007] A processing method for preventing a copy attack includes: sending, by a client, service data to a server, where the service data includes a unique identifier and a variable identifier in local secure storage of the client that are sent by the server to the client; parsing out, by the server, the unique identifier and the variable identifier in the service data sent by the client, and determining, through comparison, whether the parsed-out unique identifier and variable identifier are the same as a unique identifier and a variable identifier that are recorded by the server for the client; if a comparison result is no, determining, by the server, that the local secure storage of the client is under a copy attack, and performing a predetermined response action; or if a comparison result is yes, sending, by the server, a new variable identifier to the client; and updating, by the client, the received new variable identifier to the local secure storage.

[0008] A processing method for preventing a copy attack includes the following:

receiving service data sent by a client, where the service data includes a unique identifier and a variable identifier in local secure storage of the client that are sent by a server to the client; parsing out the unique identifier and the variable identifier in the service data sent by the client, and determining, through comparison, whether the
5 parsed-out unique identifier and variable identifier are the same as a recorded unique identifier and variable identifier of the client; and if a comparison result is no, determining that local secure storage of the client is under a copy attack, and performing a predetermined response action; or if a comparison result is yes, sending a new variable identifier to the client.

10 **[0009]** A processing method for preventing a copy attack includes the following: when local secure storage is initialized, storing a unique identifier and a variable identifier that are sent by a server; sending service data to the server, where the service data includes the unique identifier and the variable identifier in the local secure storage; receiving a new variable identifier sent by the server, where the new
15 variable identifier is a variable identifier sent by the server when the server determines through comparison that the unique identifier and the variable identifier that are uploaded by a client are the same as a unique identifier and a variable identifier that are recorded by the server for the client; and updating the received new variable identifier to the local secure storage.

20 **[0010]** A server for preventing a copy attack includes: a data receiving module, configured to receive service data sent by a client, where the service data includes a unique identifier and a variable identifier in local secure storage of the client that are sent by the server to the client; a comparison module, configured to parse out the unique identifier and the variable identifier in the service data sent by the client, and
25 determine, through comparison, whether the parsed-out unique identifier and variable identifier are the same as a recorded unique identifier and variable identifier of the client; and a processing module, configured to determine that the local secure storage of the client is under a copy attack, and perform a predetermined response action, if a comparison result of the comparison module is no; or if a comparison result of the
30 comparison module is yes, send a new variable identifier to the client.

[0011] A client for preventing a copy attack includes: a secure storage module, configured to store a unique identifier and a variable identifier that are sent by a server, when local secure storage is initialized; a data sending module, configured to send service data to the server, where the service data includes the unique identifier and the

variable identifier that are stored in the secure storage module; a first receiving module, configured to receive a new variable identifier sent by the server, where the new variable identifier is a variable identifier sent by the server when the server determines through comparison that the unique identifier and the variable identifier that are uploaded by the client are the same as a unique identifier and a variable identifier that are recorded by the server for the client; and an updating module, configured to update the received new variable identifier to the local secure storage.

[0012] The processing method for preventing a copy attack, the server, and the client provided in the present application can be used to prevent an attacker from launching a copy attack on conventional device binding. In addition, a remote server is further used for determining whether the local secure storage applied in the client is under a copy attack. In practice, the attacker obtains device data, deploys an environment, and launches an attack. These actions usually lag behind an action of reusing a terminal application by the user. In the implementations provided in the present application, time validity of stored data is set to ensure that information data stolen through a copy attack by the attacker easily expires. In addition, the server determines whether a copy attack is launched. Considering that data storage security and data computing security of the server usually have a higher priority than those of the client, it is difficult for the attacker to bypass authentication protection of the server. As such, difficulty in launching a copy attack can be increased, overall harm of implementing a copy attack can be reduced, security of data storage applied in a terminal device can be improved, a user property loss can be reduced, and user property security can be protected.

BRIEF DESCRIPTION OF DRAWINGS

[0013] To describe the technical solutions in the implementations of the present application or in the existing technology more clearly, the following briefly describes the accompanying drawings required for describing the implementations or the existing technology. Apparently, the accompanying drawings in the following description merely show some implementations of the present application, and a person of ordinary skill in the art can still derive other drawings based on these accompanying drawings without creative efforts.

[0014] FIG. 1 is a method flowchart illustrating a processing method for

preventing a copy attack, according to an implementation of the present application;

[0015] FIG. 2 is a schematic diagram illustrating an application scenario of initializing soft secure storage of a client, according to an implementation of the present application;

5 [0016] FIG. 3 is a method flowchart illustrating a processing method for preventing a copy attack, according to another implementation of the present application;

[0017] FIG. 4 is a method flowchart illustrating a processing method for preventing a copy attack, according to another implementation of the present application;

10 [0018] FIG. 5 is a schematic structural diagram illustrating modules of a server for preventing a copy attack, according to an implementation of the present application;

[0019] FIG. 6 is a schematic structural diagram illustrating modules of a server for preventing a copy attack, according to another implementation of the present application;

[0020] FIG. 7 is a schematic structural diagram illustrating modules of a client for preventing a copy attack, according to an implementation of the present application; and

20 [0021] FIG. 8 is a schematic structural diagram illustrating modules of a client for preventing a copy attack, according to another implementation of the present application.

DESCRIPTION OF IMPLEMENTATIONS

[0022] To make a person skilled in the art better understand the technical solutions in the present application, the following clearly and comprehensively describes the technical solutions in the implementations of the present application with reference to the accompanying drawings in the implementations of the present application. Apparently, the described implementations are merely some but not all of the implementations of the present application. Other implementations obtained by a person of ordinary skill in the art based on the implementations of the present application without creative efforts shall fall within the protection scope of the present application.

[0023] FIG. 1 is a method flowchart illustrating a processing method for preventing a copy attack, according to an implementation of the present application. Although the present application provides operation steps of a method or a structure of an apparatus shown in the following implementations or the accompanying
5 drawings, the method or the apparatus can include more operation steps or module units, or fewer operation steps or module structures based on conventional or non-creative efforts. For steps or structures without necessary logical causal relationship, an execution sequence of the steps or a module structure of the apparatus is not limited to an execution sequence or a module structure provided in the
10 implementations of the present application. When the method or the module structure is applied in an actual apparatus or an actual terminal product, the method or the module structure can be executed based on the sequence of the method or the module structure in the implementations or the accompanying drawings, or can be executed in parallel (for example, an environment of parallel processors, multithreaded processing,
15 or even distributed processing).

[0024] As shown in FIG. 1, a processing method for preventing a copy attack according to an implementation of the present application can include the following steps.

[0025] S1. A client sends service data to a server, where the service data includes
20 a unique identifier and a variable identifier in local secure storage of the client that are sent by the server to the client.

[0026] When the client exchanges information with the server and sends the service data to the server, the client can simultaneously send the unique identifier and the variable identifier in the local secure storage of the client to the server. The local
25 secure storage of the client can include soft secure storage of an application in the client. Generally, when local secure storage is initialized in an application installed in the client, the client can apply to the server for a unique identifier used to identify the client and a variable random code for the initialization. The unique identifier is usually used to identify an identity of the client. Generally, when it is determined that
30 the client is not under a copy attack, the unique identifier can not to change, and the variable random code can be set to be updated each time the client is triggered to use a service related to the local secure storage. Data formats of the unique identifier and the variable random code in the present application can be designed based on an actual application scenario. For example, the unique identifier in the present

implementation can include a unique ID evolving from device information of the client. The variable identifier can be a random number that is randomly generated by the server each time, for example, a six-digit combination of numerals, letters, or symbols.

5 **[0027]** In another implementation of the method in the present application, encryption processing can be performed on the unique identifier and the variable identifier that are sent by the client to the server, to further improve data storage security, and increase difficulty in data cracking. For example, a data communication process can use RSA asymmetric encryption, AES symmetric encryption, etc. that
10 includes a public key and a private key. Certainly, symmetric or asymmetric encryption can be performed on the unique identifier or the variable identifier that is delivered by the server to the client, to improve security of communication between the client and the server. Therefore, in another implementation of the processing method for preventing a copy attack in the present application, information exchange
15 of the unique identifier and the variable identifier between the client and the server can be implemented by using any encryption method of asymmetric encryption or symmetric encryption.

[0028] Certainly, when receiving the unique identifier or the variable identifier sent by the server, the client can perform decoding by using a corresponding
20 decryption algorithm, and store the obtained unique identifier or the obtained variable identifier in the local secure storage.

[0029] FIG. 2 is a schematic diagram illustrating an application scenario of initializing soft secure storage of a client, according to an implementation of the present application. In FIG. 2, when the soft secure storage of the client is initialized,
25 the client applies to a secure service of the server for a unique ID and an initial random number. The server generates and encrypts the unique ID and the initial random number of the client, and then delivers an encrypted unique ID and an encrypted initial random number to the client. The client performs decryption and securely keeps the unique ID and the initial random number.

30 **[0030]** It is worthwhile to note that in an implementation scenario in which the service data in the present implementation includes the unique identifier and the variable identifier in the local secure storage of the client that are sent by the server to the client, information triggered by the client and including the service data, the unique identifier, the variable identifier, etc. is used as the entirety of the service data

sent to the server. For example, some fields are added to the service data. The fields include the unique identifier and the variable identifier. Alternatively, in another application scenario, the service data, the unique identifier, and the variable identifier are used as separate data, and the three pieces of data are sent to the server together.

5 Certainly, the service data can be a specific service operation, or can be message information such as an authentication request of the client. Therefore, it can be understood to some extent that the message sent by the client to the server includes the unique identifier and the variable identifier of the client. Implementations are not limited in the present application.

10 **[0031]** In the present implementation of the present application, when the service data is sent to the server by using a service related to the local secure storage of the client, the service data can include the unique identifier and the variable identifier in the local secure storage of the client that are sent by the server to the client.

[0032] S2. The server parses out the unique identifier and the variable identifier in
15 the service data sent by the client, and determines, through comparison, whether the parsed-out unique identifier and variable identifier are the same as a unique identifier and a variable identifier that are recorded by the server for the client.

[0033] In the present implementation, after receiving the service data sent by the client, the server can parse the service data to obtain the unique identifier and the
20 variable identifier in the service data uploaded by the client. Because the server pre-stores the unique identifier and the previous variable identifier that are allocated to the client, the server can determine, through comparison, whether the parsed-out unique identifier and variable identifier are the same as the unique identifier and the variable identifier that are recorded and stored on the server for the client. Certainly,
25 as described above, if encryption processing is performed on the unique identifier and the variable identifier uploaded by the client, the server uses a decryption method corresponding to the encryption processing to perform decryption processing, and obtains the unique identifier and the variable identifier.

[0034] The server receives the service data sent by the client, obtains the unique
30 identifier and the variable identifier of the client, and determines, through comparison, whether the parsed-out unique identifier and variable identifier are the same as the unique identifier and the variable identifier that are recorded by the server for the client.

[0035] S3. If a comparison result is no, the server determines that local secure

storage of the client is under a copy attack, and performs a predetermined response action; or if a comparison result is yes, the server sends a new variable identifier to the client.

[0036] The server compares the parsed-out unique identifier and variable
5 identifier with the unique identifier and the variable identifier that are recorded by the server for the client. If a comparison result is that at least one of the parsed-out unique identifier and variable identifier is different from the unique identifier and the variable identifier that are recorded by the server for the client, it can indicate that the client that sends the service data is abnormal. Particularly, if the variable identifiers, such as
10 the random numbers in the present implementation, are different, it indicates that the corresponding soft secure storage applied in the client probably has been under a copy attack and the application has been run, thereby triggering a related service of the soft secure storage. Therefore, in the present implementation of the present application, if the server determines through comparison that the parsed-out unique identifier and
15 variable identifier are different from the unique identifier and the variable identifier that are recorded by the server for the client, the server can determine that the local secure storage of the client is under a copy attack, and can further perform the predetermined response action. The response action can be predetermined based on an actual situation, for example, forced logout, prohibition on a financial operation of a
20 user account, and encryption protection authentication.

[0037] Certainly, if the server determines through comparison that the parsed-out unique identifier and variable identifier are the same as the unique identifier and the variable identifier that are recorded by the server for the client, it can be considered that the client that sends a service request is an authentic client of the user. Service
25 access can be allowed, and a further operation can be performed. Then, the server can generate a new variable identifier for the client, and then send the new variable identifier to the client. In the present implementation, if the server determines that the unique ID and the random number that are uploaded by the client are consistent with the unique ID and the previously sent random number that are recorded in the storage
30 for the client, identity authentication of the client succeeds, and the server generates a new random number and sends the new random number to the client.

[0038] S4. The client updates the received new variable identifier in the local secure storage.

[0039] As described above, when the server determines that the unique identifier

and the variable identifier uploaded by the client are the same as the unique identifier and the variable identifier that are recorded by the server for the client, the server can send the new variable identifier to the client, and the client can update the new variable identifier in the local secure storage of the client for use by service data to be sent for the next time.

[0040] The processing method for preventing a copy attack provided in the present application can be used to prevent an attacker from launching a copy attack on regular device binding. In addition, a remote server is further used for determining whether the local secure storage applied in the client is under a copy attack. In a real use environment, the attacker obtains device data, deploys an attacking environment, and launches an attack. These actions usually lag behind an action of reusing a terminal application by the user. In the implementations provided in the present application, time validity of stored data is set to ensure that information data stolen through a copy attack by the attacker easily expires. In addition, the server determines whether a copy attack is launched. Considering that data storage security and data computing security of the server usually have a higher priority than those of the client, it is difficult for the attacker to bypass authentication protection of the server. As such, difficulty in launching a copy attack can be increased, overall harm of implementing a copy attack can be reduced, security of data storage applied in a terminal device can be improved, a user property loss can be reduced, and user property security can be protected.

[0041] Certainly, as described above, if a comparison result is no, i.e., if the comparison result is that the server determines that at least one of the parsed-out unique identifier and variable identifier is different from the unique identifier and the variable identifier that are recorded by the server for the client, it can be considered that the local secure storage of the client is under a copy attack. Further, the response action can be performed. The response action in the present application can be set based on an actual situation, for example, forced logout, prohibition on a financial operation of a user account, and encryption protection authentication. When it is determined that the local secure storage of the client is under a copy attack, a processing method is provided based on the method in the present application. In another implementation of the processing method for preventing a copy attack in the present application, the predetermined response action can include the following step:

[0042] S301. The server delivers a message that identity authentication needs to

be performed on a user corresponding to the service data, and after the identity authentication succeeds, sends a redistributed unique identifier and a redistributed variable identifier to the client of the user.

[0043] Correspondingly, the client of the user performs identity authentication based on the received identity authentication message, for example, password login, fingerprint login, and face login. After the user passes the identity authentication on the client of the user, the local secure storage can be initialized based on the received unique identifier and the received variable identifier that are redistributed by the server for the client of the user.

[0044] The variable identifier in the present application can use one or a combination of numerals, letters, or symbols. In an implementation of the variable identifier provided in the present application, time validity of the variable identifier (such as a random integer) can be implemented through integer accumulation. Therefore, in an implementation of the method in the present application, the variable identifier is set to be generated through successive integer accumulation.

[0045] In an application scenario, the variable identifier of the client C1 uses an initial random six-digit integer 013579. Each time a service of the secure storage triggers update of the variable identifier, an updated variable identifier is generated by adding 1 to the value of the previous variable identifier. For example, the initial variable identifier is 013579, a variable identifier generated and sent by the server to the client C1 for the next time is 013580, 013581 for next time, and so on. Certainly, the server can randomly generate different initial random six-digit integers for different clients.

[0046] The processing method for preventing a copy attack can be applied to a security authentication server. Time validity is set for data to ensure that data stolen through a copy attack by a hacker easily expires. In addition, the server determines whether the client is under a copy attack. Therefore, difficulty in launching a copy attack is increased, harm of implementing a copy attack is reduced, and security of data storage applied in the client is improved. Therefore, based on the implementations described above, the present application further provides a processing method for preventing a copy attack that can be applied to a server. The method can include:

[0047] S201. Receive service data sent by a client, where the service data includes a unique identifier and a variable identifier in local secure storage of the client that are

sent by a server to the client.

[0048] S202. Parse the unique identifier and the variable identifier in the service data sent by the client, and determine, through comparison, whether the parsed-out unique identifier and variable identifier are the same as a recorded unique identifier and variable identifier of the client.

[0049] S203. If a comparison result is no, determine that the local secure storage of the client is under a copy attack, and perform a predetermined response action; or if a comparison result is yes, send a new variable identifier to the client.

[0050] FIG. 3 is a method flowchart illustrating a processing method for preventing a copy attack, according to an implementation of the present application. In an application scenario, when a user uses a service related to soft secure storage, a client encrypts a unique ID and a random number in the soft secure storage, and sends encrypted unique ID and encrypted random number together with service data to a server. The server obtains the unique ID and the random number through decryption.

If it is determined that the unique ID and the random number are consistent with a unique ID and a random number that are stored on the server, it indicates that the soft secure storage is not under a copy attack. The server updates a new random number for the soft secure storage, and the soft secure storage updates the local random number.

[0051] In another application scenario, if the server finds that the random number uploaded by the client is different from a random number recorded by the server, it may indicate that the soft secure storage has been under a copy attack and run. In this case, the server can perform identity authentication on a user of an account and perform initialization. In another implementation of the method of the present application, when the server determines that the local secure storage of the client is under a copy attack, a predetermined response action can include: delivering a message that identity authentication needs to be performed on a user corresponding to the service data, and after the identity authentication succeeds, sending, to the client of the user, a redistributed unique identifier and a redistributed variable identifier.

[0052] The variable identifier can be set to be generated through successive integer accumulation. In addition, in another implementation, information exchange of the unique identifier and the variable identifier between the client and the server can be implemented by using any encryption method of asymmetric encryption or symmetric encryption. For details of methods for data setting, data generation, and

data exchange, reference can be made to descriptions of other implementations of the present application. Details are omitted here for simplicity.

[0053] The present application provides a processing method for preventing a copy attack that can be applied to a server, so that the server determines whether the client is under a copy attack. A corresponding action is performed based on a determining result, to ensure as much as possible that user data stolen by a hacker easily expires. Therefore, security of data storage of the client is improved, and a processing method for preventing a copy attack and reducing a loss caused by a copy attack on a user is provided.

[0054] Currently, the processing method for preventing a copy attack in the present application is applicable to a client. Local secure storage of the client stores a unique identifier that is configured by a server for the client and a variable identifier that is updated each time a service operation is performed. When the client interacts with the server, the client sends the local identifier and the variable identifier to the server, and then performs a local operation based on the result of determining by the server, for example, account identity authentication, soft secure storage initialization, and random number updating. The present application provides the processing method for preventing a copy attack that can be applied to a client. The method can include the following steps:

[0055] S301. When local secure storage is initialized, store a unique identifier and a variable identifier that are sent by a server.

[0056] S302. Send service data to the server, where the service data includes the unique identifier and the variable identifier in the local secure storage.

[0057] S303. Receive a new variable identifier sent by the server, where the new variable identifier is a variable identifier sent by the server, when the server determines through comparison that the unique identifier and the variable identifier that are uploaded by the client are the same as a unique identifier and a variable identifier that are recorded by the server for the client.

[0058] S304. Update the received new variable identifier to the local secure storage.

[0059] FIG. 4 is a method flowchart illustrating a processing method for preventing a copy attack, according to another implementation of the present application. Certainly, in another implementation, the processing method for preventing a copy attack that can be applied to a client can further include: performing

identity authentication based on a received identity authentication message, and after the identity authentication succeeds, initializing the local secure storage based on a received unique identifier and variable identifier that are redistributed by the server.

[0060] The server, and the client, and the processing method performed between

5 the server and the client for preventing a copy attack, provided in the implementations of the present application, can be used to prevent an attacker from launching a copy attack on conventional device binding. In addition, a remote server is further used for determining whether the local secure storage applied in the client is under a copy attack. In a real use environment, the attacker obtains device data, deploys an
10 environment, and launches an attack. These actions usually lag behind an action of reusing a terminal application by the user. In the implementations provided in the present application, time validity of stored data is set to ensure that information data stolen through a copy attack by the attacker easily expires. In addition, the server determines whether a copy attack is launched. Considering that data storage security
15 and data computing security of the server usually have a higher priority than those of the client, it is difficult for the attacker to bypass authentication protection of the server. As such, difficulty in launching a copy attack can be increased, overall harm of implementing a copy attack can be reduced, security of data storage applied in a terminal device can be improved, a user property loss can be reduced, and user
20 property security can be protected.

[0061] Based on the processing method for preventing a copy attack described in the previous implementations, the present application further provides a processing apparatus for preventing a copy attack. The apparatus can be applied in a server, and a system using a secure service. FIG. 5 is a schematic structural diagram illustrating
25 modules of a server for preventing a copy attack, according to an implementation of the present application. As shown in FIG. 5, the server can include: a data receiving module 101, configured to receive service data sent by a client, where the service data includes a unique identifier and a variable identifier in local secure storage of the client that are sent by the server to the client; a comparison module 102, configured to
30 parse out the unique identifier and the variable identifier in the service data sent by the client, and determine, through comparison, whether the parsed-out unique identifier and variable identifier are the same as a recorded unique identifier and variable identifier of the client; and a processing module 103, configured to determine that the local secure storage of the client is under a copy attack, and perform a

predetermined response action if a comparison result of the comparison module 102 is no; or if a comparison result of the comparison module 102 is yes, send a new variable identifier to the client.

[0062] The predetermined response action can be set based on a specific application scenario. In an implementation provided in the present application, if it is determined that the local secure storage of the client is under a copy attack, identity authentication can be performed again on a user, and the local secure storage can be initialized. FIG. 6 is a schematic structural diagram illustrating modules of a processing server for preventing a copy attack, according to another implementation of the present application. As shown in FIG. 6, the processing module 103 can include: a first processing module 1031, configured to deliver a message that identity authentication needs to be performed on a user corresponding to the service data when it is determined that the local secure storage of the client is under a copy attack, and after the identity authentication succeeds, send, to the client of the user, a redistributed unique identifier and a redistributed variable identifier; and a second processing module 1032, configured to send, by the server, a new variable identifier to the client, when it is determined that the local secure storage of the client is not under a copy attack.

[0063] As described above, in another implementation of the apparatus in the present application, the variable identifier is set to be generated through successive integer accumulation. In addition, information exchange of the unique identifier and the variable identifier between the client and the server is implemented by using any encryption method of asymmetric encryption or symmetric encryption. For details, reference can be made to related descriptions in other implementations of the present application. Details are omitted here for simplicity.

[0064] The processing server for preventing a copy attack described above can determine whether the local secure storage applied in the client is under a copy attack. In a real use environment, the attacker obtains device data, deploys an environment, and launches an attack. These actions usually lag behind an action of reusing a terminal application by the user. In the implementations provided in the present application, time validity of stored data is set to ensure that information data stolen through a copy attack by the attacker easily expires. In addition, the server determines whether a copy attack is launched. Considering that data storage security and data computing security of the server usually have a higher priority than those of the client,

it is difficult for the attacker to bypass authentication protection of the server. As such, difficulty in launching a copy attack can be increased, overall harm of implementing a copy attack can be reduced, security of data storage applied in a terminal device can be improved, a user property loss can be reduced, and user property security can be protected.

[0065] The present application further provides a client for preventing a copy attack. During information exchange with a server, whether local secure storage of a client is under a copy attack can be determined based on a unique identifier and a variable identifier that are allocated by the server. Methods such as changing a random number are applied to ensure that data stolen through a copy attack by an attacker easily expires. The server determines whether the client is under a copy attack. Therefore, data in the local secure storage of the client is effectively protected, data storage security is improved, and difficulty in launching a copy attack is increased. FIG. 7 is a schematic structural diagram illustrating modules of a client for preventing a copy attack, according to an implementation of the present application. As shown in FIG. 7, the client can include: a secure storage module 201, configured to store a unique identifier and a variable identifier that are sent by a server when local secure storage is initialized; a data sending module 202, configured to send service data to the server, where the service data includes the unique identifier and the variable identifier that are stored in the secure storage module 201; a first receiving module 203, configured to receive a new variable identifier sent by the server, where the new variable identifier is a variable identifier sent by the server when the server determines through comparison that the unique identifier and the variable identifier that are uploaded by the client are the same as a unique identifier and a variable identifier that are recorded by the server for the client; and an updating module 204, configured to update the received new variable identifier to the local secure storage.

[0066] FIG. 8 is a schematic structural diagram illustrating modules of a client for preventing a copy attack, according to another implementation of the present application. As shown in FIG. 8, the client can further include: a second receiving module 2051, configured to receive an identity authentication message sent by the server, and receive a redistributed unique identifier and a redistributed variable identifier that are sent by the server after identity authentication succeeds; and an authentication processing module 2052, configured to perform identity authentication based on the received identity authentication message, and after the identity

authentication succeeds, initialize the local secure storage based on the received unique identifier and the received variable identifier that are redistributed by the server.

[0067] The present application provides the client for preventing a copy attack.

5 During information exchange with a server, whether local secure storage of a client is under a copy attack can be determined based on a unique identifier and a variable identifier that are allocated by the server. Methods such as changing a random number are applied to ensure that data stolen through a copy attack by an attacker easily expires. The server determines whether the client is under a copy attack. Therefore,
10 data in the local secure storage of the client is effectively protected, data storage security is improved, and difficulty in launching a copy attack is increased.

[0068] The method, the server, or the client described in the previous implementations can implement secure storage in a trusted execution environment (TEE) with relatively good effects. Certainly, in another execution environment such
15 as an ANDROID system environment, a language such as c/c++, a virtual machine technology, and a technology such as obfuscation can be used to implement secure storage, thereby improving security of soft secure storage.

[0069] The content of the present application includes descriptions of data storage, setting, and an information exchange method, such as local secure storage of data, a
20 data format of a variable identifier, encryption/decryption processing, and information data sending/receiving. However, the present application is not limited to situations satisfying industry processing standards, specifications, or implementations. A slightly modified implementation based on some industry standards or implementations can also achieve expected implementation effects same as, equivalent to, or close to those
25 achieved based on the previous implementations or variants of the previous implementations. Applying these modified or changed data storage, setting, an information exchange method, etc. can still fall within a protection scope of an optional implementation of the present application.

[0070] Although the present application provides method operation steps
30 described in the implementations or flowcharts, more or fewer operation steps can be included based on conventional or non-creative means. The sequence of the steps listed in the implementations is merely one of numerous step execution sequences, and does not represent the unique execution sequence. For an actual apparatus or client product, the steps can be executed based on the method sequence illustrated in

the implementations or accompanying drawings or executed in parallel (for example, an environment of parallel processors or multi-threaded processing).

5 [0071] The units, apparatuses, or modules described in the previous implementations can be implemented by a computer chip or an entity, or implemented by a product with a certain function. For ease of description, the previous apparatus is described by dividing the functions into various modules. Certainly, when the present application is implemented, the functions of the modules can be implemented by using the same or multiple pieces of software and/or hardware, or a module for implementing a function is implemented by using a combination of multiple submodules or subunits.

10 [0072] A person skilled in the art also knows that, in addition to implementing a controller by using a computer readable program code, logic programming can be performed on method steps to allow the controller to implement the same function in forms of a logic gate, a switch, an application-specific integrated circuit, a programmable logic controller, an embedded microcontroller, etc. Therefore, the controller can be considered as a hardware component, and an apparatus included in the controller and configured to implement various functions can also be considered as a structure in the hardware component. Alternatively, the apparatus configured to implement various functions can even be considered as both a software module implementing the method and a structure in the hardware component.

20 [0073] The present application can be described in the general context of computer executable instructions executed by a computer, for example, a program module. Generally, the program module includes a routine, a program, an object, a component, a data structure, a type, etc. that executes a specific task or implements a specific abstract data type. The present application can also be practiced in distributed computing environments. In the distributed computing environments, tasks are performed by remote processing devices connected through a communications network. In a distributed computing environment, the program module can be located in both local and remote computer storage media that including storage devices.

30 [0074] It can be seen from the descriptions of the implementations that a person skilled in the art can clearly understand that the present application can be implemented by using software and a necessary general hardware platform. Based on such an understanding, the technical solutions of the present application essentially or the part contributing to the existing technology can be implemented in a form of a

software product. The software product can be stored in a storage medium such as a read-only memory (ROM)/random access memory (RAM), a magnetic disk, or an optical disc, and includes several instructions for instructing a computer device (which can be a personal computer, a mobile device, a server, a network device, etc.)

5 to perform the methods described in the implementations or in some parts of the implementations of the present application.

[0075] The implementations in the present specification are described in a progressive manner. For the same or similar parts in the implementations, reference can be made to each other. Each implementation focuses on a difference from other implementations. The present application can be applied in many general-purpose or dedicated computer system environments or configurations, for example, a personal computer, a server computer, a handheld device or a portable device, a tablet device, a multi-processor system, a microprocessor-based system, a set top box, a programmable video playback system, a network PC, a small computer, a mainframe computer, and a distributed computing environment including any one of the previous systems or devices.

[0076] Although the present application is described by using the implementations, a person of ordinary skill in the art knows that many modifications and variations of the present application can be made without departing from the spirit of the present application. It is expected that the claims include these modifications and variations without departing from the spirit of the present application.

CLAIMS

1. A processing method for preventing an attacker from successfully launching a copy attack, wherein the method comprises:

sending, by a client, service data to a server, wherein:

the service data comprises a unique identifier and a variable identifier stored in local secure storage of the client that are sent by the server to the client, and

the variable identifier is generated by the server;

parsing, by the server, the service data sent by the client to extract the unique identifier and the variable identifier;

determining, through comparison, whether the extracted unique identifier and variable identifier are the same as a unique identifier and variable identifier that are recorded by the server for the client;

in response to determining that the extracted unique identifier and variable identifier are not the same as the unique identifier and variable identifier that are recorded by the server for the client:

determining, by the server, that the local secure storage of the client is under a copy attack comprising an attempt to generate an unauthorized data copy, and

performing a predetermined response action; or

in response to determining that the extracted unique identifier and variable identifier are the same as the unique identifier and variable identifier that are recorded by the server for the client:

sending, by the server, a new variable identifier to the client, and updating, by the client, the received new variable identifier to the local secure storage.

2. The processing method according to claim 1, wherein the predetermined response action comprises:

delivering, by the server, a message that indicates that identity authentication needs to be performed on a user corresponding to the service data; and

after the identity authentication succeeds, sending, to the client of the user, a redistributed unique identifier and a redistributed variable identifier.

3. The processing method according to claim 1 or 2, wherein the variable identifier is generated through successive integer accumulation.
4. The processing method according to claim 1 or 2, wherein information exchange of the unique identifier and the variable identifier between the client and the server is implemented using any encryption method of asymmetric encryption or symmetric encryption.
5. The processing method according to claim 1, wherein the method further comprises, before the step of sending, by the client, service data to the server:
 - initializing the local secure storage, wherein initializing the local secure storage comprises:
 - applying, by the client and to the server, for the unique identifier used to identify the client and a variable random code, and
 - in response to local secure storage being initialized, storing a unique identifier and a variable identifier that are sent by the server.
6. The processing method according to claim 5, wherein the method further comprises:
 - performing identity authentication based on a received identity authentication message;
 - and
 - after the identity authentication succeeds, initializing the local secure storage based on a received unique identifier and variable identifier that are redistributed by the server.
7. The processing method according to any one of claims 1 to 6, wherein the unique identifier identifies an identity of the client and the variable identifier comprises a random number that is randomly generated by the server each time a variable identifier is sent to the client.
8. A server for preventing an attacker from successfully launching a copy attack, wherein the server comprises:
 - a data receiving module, configured to receive service data sent by a client, wherein:
 - the service data comprises a unique identifier and a variable identifier stored in local secure storage of the client that are sent by the server to the client, and
 - the variable identifier is generated by the server;

a comparison module, configured to parse the service data sent by the client to extract the unique identifier and the variable identifier in the service data sent by the client, and determine, through comparison, whether the extracted unique identifier and variable identifier are the same as a recorded unique identifier and variable identifier of the client; and

a processing module, configured to: if the extracted unique identifier and variable identifier are not the same as the recorded unique identifier and variable identifier, determine that the local secure storage of the client is under a copy attack comprising an attempt to generate an unauthorized data copy, and perform a predetermined response action; or if the extracted unique identifier and variable identifier are the same as the recorded unique identifier and variable identifier, send a new variable identifier to the client.

9. The server according to claim 8, wherein the processing module comprises:

a first processing module, configured to in response to a determination that the local secure storage of the client is under a copy attack, deliver a message that indicates that identity authentication needs to be performed on a user corresponding to the service data, and after the identity authentication succeeds, send, to the client of the user, a redistributed unique identifier and a redistributed variable identifier; and

a second processing module, configured to in response to a determination that the local secure storage of the client is not under a copy attack, send, by the server, a new variable identifier to the client.

10. The server according to claim 8 or 9, wherein the variable identifier is set to be generated through successive integer accumulation.

11. The server according to claim 8 or 9, wherein information exchange of the unique identifier and the variable identifier between the client and the server is implemented using any encryption method of asymmetric encryption or symmetric encryption.

12. The server according to any one of claims 8 to 11, wherein the server is further configured to:

receive, from the client, an application for the unique identifier and a variable random code,

wherein the unique identifier is used to identify the client; and
send the unique identifier and the variable identifier to the client.

13. A client for preventing an attacker from successfully launching a copy attack, wherein the client comprises:

a secure storage module, configured to, in response to local secure storage being initialized, store a unique identifier and a variable identifier that are sent by a server, wherein the variable identifier is generated by the server;

a data sending module, configured to send service data to the server, wherein the service data comprises the unique identifier and the variable identifier that are stored in the secure storage module;

a first receiving module, configured to receive a new variable identifier sent by the server, wherein the new variable identifier is a variable identifier sent by the server in response to the server determining through a comparison, that the unique identifier and the variable identifier that are uploaded by the client are the same as a unique identifier and a variable identifier that are recorded by the server for the client; and

an updating module, configured to update the received new variable identifier to the local secure storage.

14. The client according to claim 13, wherein the client further comprises:

a second receiving module, configured to receive an identity authentication message sent by the server, and receive a redistributed unique identifier and a redistributed variable identifier that are sent by the server after identity authentication succeeds; and

an authentication processing module, configured to perform identity authentication based on the received identity authentication message, and after the identity authentication succeeds, initialize the local secure storage based on the received unique identifier and the received variable identifier that are redistributed by the server.

15. The client according to claim 13, wherein the client is further configured to apply, to the server, for the unique identifier used to identify the client and a variable random code.