

LOGIN TOKEN GENERATION METHOD AND APPARATUS, LOGIN TOKEN VERIFICATION METHOD AND APPARATUS, AND SERVER

This application claims priority to Chinese Patent Application No. 201910691438.1, filed with the National Intellectual Property Administration, PRC on July 29, 2019 and entitled "LOGIN TOKEN GENERATION METHOD AND APPARATUS, LOGIN TOKEN VERIFICATION METHOD AND APPARATUS, AND SERVER", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

This application relates to the field of information technologies, and specifically to a login token generation method and apparatus, a login token verification method and apparatus, and a server.

BACKGROUND

A login token is a string generated by a serving end, and is used as an identifier used by a client to initiate a request. When a user logs in for the first time, the serving end generates a token and returns the token to the client. Thereafter, when the client initiates a request, it is only required to add the token to the request, and there is no need to enter a username and a password again.

In the prior art, the serving end usually generates a login token based on a username and a password used by a user to log in. This generation method is too simple, and the generated token can be stolen easily because only the username and the password need to be verified. In addition, the token remains unchanged within a time period after the token is generated, and when initiating a new request to the serving end, the user can continue to perform an operation by using the original token even if user information and permission of the user have changed, resulting in low security.

SUMMARY

One of the objectives of embodiments of this application is to provide a login token generation method and apparatus, a login token verification method and apparatus, and a server, to resolve problems in the prior art that a login token generation method is too simple, a login token can be stolen easily, and security is relatively low.

To resolve the foregoing technical problems, the technical solutions used in the embodiments of this application are as follows:

According to a first aspect, a login token generation method is provided, and includes:

receiving a login request submitted by a user by using a browser, and verifying a user ID and a password that are included in the login request;

obtaining browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

concatenating the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext based on the user ID and a preset token expiration time, and encrypting the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

encrypting a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, where the header is obtained by encrypting preset header plaintext by using the first encryption algorithm; and

combining the header, the raw data, and the dynamic key into a login token.

According to a second aspect, a login token verification method is provided, and includes:

receiving an operation request submitted by a user by using a browser, where the operation request includes a login token issued previous time;

extracting a user ID of the user from the login token;

obtaining user information of the user corresponding to the user ID and browser information of the browser;

concatenating the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypting the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

encrypting a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, where the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

combining the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

verifying the login token by using the verification token.

According to a third aspect, a login token generation apparatus is provided, and includes:

a login request receiving module, configured to receive a login request submitted by a user by using a browser, and verify a user ID and a password that are included in the login request;

an information obtaining module, configured to obtain browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

a key plaintext concatenation module, configured to concatenate the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

a raw data encryption module, configured to generate raw data plaintext based on the user ID and a preset token expiration time, and encrypt the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

a dynamic key encryption module, configured to encrypt a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, where the header may be obtained by encrypting preset header plaintext by using the first encryption algorithm; and

a login token combination module, configured to combine the header, the raw data, and the dynamic key into a login token.

According to a fourth aspect, a login token verification apparatus is provided, and includes:

a receiving module, configured to receive an operation request submitted by a user by using a browser, where the operation request includes a login token issued previous time;

an extraction module, configured to extract a user ID of the user from the login token;

an obtaining module, configured to obtain user information of the user corresponding to the user ID and browser information of the browser;

a concatenation module, configured to concatenate the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

a first encryption module, configured to generate raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypt the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

a second encryption module, configured to encrypt a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using

a preset second encryption algorithm, to obtain a dynamic key of the verification token, where the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

a combination module, configured to combine the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

a verification module, configured to verify the login token by using the verification token.

According to a fifth aspect, a server is provided, and includes a memory, a processor, and a computer-readable instruction that is stored in the memory and can run on the processor, where when the processor executes the computer-readable instruction, the login token generation method according to the first aspect is implemented.

According to a sixth aspect, a server is provided, and includes a memory, a processor, and a computer-readable instruction that is stored in the memory and can run on the processor, where when the processor executes the computer-readable instruction, the login token verification method according to the second aspect is implemented.

According to a seventh aspect, a nonvolatile computer-readable storage medium is provided, where the nonvolatile computer-readable storage medium stores a computer-readable instruction, and when the computer-readable instruction is executed by a processor, the login token generation method according to the first aspect is implemented.

According to an eighth aspect, a nonvolatile computer-readable storage medium is provided, where the nonvolatile computer-readable storage medium stores a computer-readable instruction, and when the computer-readable instruction is executed by a processor, the login token verification method according to the second aspect is implemented.

Beneficial effects of the login token generation method and apparatus, the login token verification method and apparatus, and the server provided in the embodiments of this application

are as follows: A token is generated based on user information of a requestor, browser information, an IP address, and the like, which increases login token generation complexity, and can prevent login from an unusual location and a cross-browser operation. In addition, user permission-related information is added to generate the login token, so that after user permission is changed, the corresponding login token becomes invalid immediately, and therefore a change of the user permission can be reflected, an unauthorized operation can be prevented, and login token security and system security can be improved.

BRIEF DESCRIPTION OF DRAWINGS

To describe the technical solutions in the embodiments of this application more clearly, the following briefly describes the accompanying drawings required for describing the embodiments or example technologies. Clearly, the accompanying drawings in the following description are merely some embodiments of this application, and a person of ordinary skill in the art can further derive other drawings from these accompanying drawings without creative efforts.

FIG. 1 is a schematic flowchart of steps of a login token generation method according to an embodiment of this application;

FIG. 2 is a schematic diagram of a process of generating a dynamic key according to an embodiment of this application;

FIG. 3 is a schematic flowchart of steps of a login token verification method according to an embodiment of this application;

FIG. 4 is a schematic diagram of a login token generation apparatus according to an embodiment of this application;

FIG. 5 is a schematic diagram of a login token verification apparatus according to an embodiment of this application; and

FIG. 6 is a schematic diagram of a server according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

To make the objectives, technical solutions, and advantages of this application more clearly, the following further describes this application in detail with reference to the accompanying drawings and embodiments. It should be understood that the specific embodiments described herein are merely intended to explain this application, and are not intended to limit this application.

To describe the technical solutions of this application, the following provides detailed description with reference to the specific accompanying drawings and embodiments.

FIG. 1 is a schematic flowchart of steps of a login token generation method according to an embodiment of this application. The method may specifically include the following steps.

S101. Receive a login request submitted by a user by using a browser, and verify a user ID and a password that are included in the login request.

It should be noted that the method may be applied to various systems in which the user needs to log in before performing an operation, for example, an online banking system and a securities trading system. A scenario to which the method is specifically applied is not limited in this embodiment.

Usually, for the forgoing system, after the user submits the login request by entering the user ID and the password, a serving end can generate a login token and return the login token to a client. The login token may be stored in a flash memory of the client. When subsequently performing an operation in the system, the user needs to carry only the login token, and does not need to enter verification information such as the user ID and the password again. A process of generating a login token by the serving end is described in this embodiment.

The client refers to a program that can carry components of the corresponding system and allow the user to perform operations to implement various functions. The program may be a client software program installed independently on a terminal device, or may be a system web page program provided based on a browser. This is not limited in this embodiment. The client needs to

work with the serving end. The serving end is a backend server communicatively connected to the client.

A browser program is usually embedded in the client software program installed independently on the terminal device, and a login interface can be generated by using the browser program for the user to perform a login operation. A login operation performed by the user in the independent client software program or a login operation performed by the user in the browser-based system web page program can be considered as being completed in the browser. In other words, the user can submit the login request to the serving end by using the browser.

For ease of understanding, in this embodiment, subsequent description is provided by using an example in which the client is a system web page program provided based on a browser. In other words, the user directly performs operations in the browser to implement various functions.

Usually, when submitting the login request, the user needs to enter the user ID and the password in the login interface of the system. After receiving the login request, the serving end can verify the user ID and the password to determine whether the user ID has permission to access the current system.

S102. Obtain browser information of the browser and user information of the user after the verification on the user ID and the password succeeds.

After receiving the user ID and the password, the serving end can search a database for a password corresponding to the user ID, and determine whether the password corresponding to the user ID is the same as the received password. If the password corresponding to the user ID is the same as the received password, it can be considered that the user ID has permission to access the current system. Otherwise, the login request of the user ID can be rejected.

After verifying the user ID and the password, the serving end may further obtain the browser information and the user information to generate a login token.

The browser information in this embodiment may include at least one of a name of the browser,

version information of the browser, and current network environment information of the browser, and the network environment information may include an IP address of a current network environment.

The browser information may be collected when the user logs into the system for the first time. In other words, after the user logs into the system for the first time, the system may identify a type of the browser currently used by the user, and save the browser information. Certainly, the obtained browser information may further include another type of information based on an actual requirement. This is not limited in this embodiment.

In this embodiment, in addition to the user ID, the user information may further include permission information and login/logout information of the current user. The permission information of the user includes at least one of a user status, a user level, a user role, and the latest update time of the user role, and the login/logout information may include the latest login time, the latest logout time, and the like of the user in the system. The permission information and the login/logout information may be obtained from the database of the serving end. Certainly, similar to the browser information, content included in the permission information and the login/logout information may also be specifically determined based on different actual requirements. This is not limited in this embodiment.

Both the browser information and the user information can be expressed as strings. For example, the browser information includes the name of the browser: Mozilla, the version information: 5.0, and the IP address: 127.0.0.1.

An existing login token usually remains unchanged within a time period after being generated, an unauthorized operation cannot be prevented after user information and permission are changed, and operations from an unusual location or operations simultaneously performed by multiple browsers cannot be effectively perceived. In this embodiment, the user information and the information about the currently used browser are obtained to generate a login token, and therefore system security can be effectively improved. When the user performs cross-browser login, if

verification on the user ID and the login password succeeds, information about the new browser is saved, and the operation of generating a login token continues to be performed. However, when the user returns to the original browser for operation, the original login token is invalid because the latest login time and the browser information have been updated. Therefore, in this embodiment, when the user performs a cross-browser operation, login of the original browser is invalid, to ensure that the user can perform an operation only by using the same window of the same browser in the same session. However, when the user logs in again in another session and verification succeeds, the previous session becomes invalid immediately.

S103. Concatenate the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence.

In the process of generating a login token by the serving end, the obtained user ID, user information, and browser information may be concatenated into the key plaintext based on a specific concatenation method and concatenation sequence. A dynamic key may be obtained after the key plaintext is encrypted based on a specific encryption algorithm.

Specifically, the obtained information may be separated by using a specific type of separator. For example, two pieces of different information may be separated by using a vertical line separator "|".

In an example of this application, key plaintext that uses the vertical line separator and that is obtained through concatenation based on a specific concatenation sequence may be expressed as

6|127.0.0.1|Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36|Thu Jan 31 17:51:57 CST 2019|Mon Feb 11 15:19:39 CST 2019|Mon Feb 11 14:59:09 CST 2019|1|0|63fde500-2dcd-11e9-b37b-95aef6d46d16|5|Thu Jan 31 17:43:06 CST 2019|.

In the key plaintext, each string separated by using the vertical line separator indicates one piece of specific information. For example, "6" represents the user ID, "127.0.0.1" represents the IP

address, and the subsequent strings represent other types of information that need to be used to generate a login token.

S104. Generate raw data plaintext based on the user ID and a preset token expiration time, and encrypt the raw data plaintext by using a preset first encryption algorithm, to obtain raw data.

The raw data plaintext usually includes at least the user ID and the token expiration time. The token expiration time indicates when the currently generated login token should expire, in other words, not take effect.

After being generated based on the user ID and the token expiration time, the raw data plaintext may be encrypted by using the preset first encryption algorithm. A string obtained after the encryption is used as some raw data of the login token.

The first encryption algorithm may be a Base64 algorithm. Base64 is a representation method in which binary data is represented based on 64 printable characters. Because 2 raised to the power of 6 is equal to 64, every 6 bits are used as one unit to correspond to a printable character. Three bytes include 24 bits and correspond to four Base64 units; in other words, the three bytes need to be represented by using four printable characters. Certainly, the raw data plaintext may be encrypted by using another algorithm based on a different actual requirement.

In an example, the raw data plaintext may be expressed as {"sub": "user", "iss": "oneconnect", "exp": "1549871380", "userID": "6"}, where

sub represents the user to which the login token is orientated, iss represents an issuer of the login token, exp represents the token expiration time and is a Unix timestamp herein, and userID represents the user ID.

After the raw data plaintext is encrypted by using the Base64 algorithm, the obtained raw data is eyJzdWliOiJ1c2VyliwiaXNzljoiY25Ly29ubmVjdCIsImV4cCI6MTU0OTg3MTM4MCwidXNlckklkljoiNiJ9.

S105. Encrypt a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, where the header is obtained by encrypting preset header plaintext by using the first encryption algorithm.

Usually, an algorithm used to encrypt the preset header plaintext may be the same as the algorithm used to encrypt the raw data plaintext. Therefore, the header plaintext may be encrypted by using the first encryption algorithm, namely, the Base64 algorithm, to obtain the header.

The header may be considered as a header of a login token, and is used to describe basic information about the login token, for example, a type of login token and a signature algorithm.

In an example, the header text may be expressed as {"typ": "JWT", "alg": "HS256"}, where

typ represents the type of the login token, and alg indicates that the signature algorithm is HS256.

After the header plaintext is encrypted by using Base64, the obtained header may be expressed as eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.

After the header, the raw data, and the key plaintext are obtained, the header, the raw data, and the key plaintext may be encrypted by using another encryption algorithm, namely, the second encryption algorithm, to obtain the dynamic key.

The second encryption algorithm may be an HMAC SHA256 algorithm. A hash-based message authentication code (HMAC) is a key-related hash operation message authentication code, and a hash algorithm can be operated to generate a message digest as an output by using a key and a message as inputs. HMAC includes many hash encryption algorithms, and HMAC SHA256 is one of the hash encryption algorithms.

FIG. 2 is a schematic diagram of a process of generating a dynamic key according to an embodiment of this application. Based on the process shown in FIG. 2, the header plaintext and the raw data plaintext may be encrypted by using the same Base64 algorithm, and two obtained strings

are used as the header and the raw data. Then, the header, the raw data, and the key plaintext may be encrypted by using the HMAC SHA256 algorithm, to obtain the dynamic key.

In an example of this embodiment, after the header eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9, the raw data eyJzdWliOiJ1c2VyliwiaXNzljoiI25Ly29ubmVjdCIsImV4cCI6MTU0OTg3MTM4MCwidXNlckkljoiNiJ9, and the key plaintext 6|127.0.0.1|Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36|Thu Jan 31 17:51:57 CST 2019|Mon Feb 11 15:19:39 CST 2019|Mon Feb 11 14:59:09 CST 2019|1|0|63fde500-2dcd-11e9-b37b-95aef6d46d16|5|Thu Jan 31 17:43:06 CST 2019| are encrypted by using the HMAC SHA256 algorithm, the obtained dynamic key may be expressed as PwXj8KoRbRN5iJFdNopiOTcinBIvejve9myJgKI8FC8.

S106. Combine the header, the raw data, and the dynamic key into a login token.

A login token usually includes a header, raw data, and a dynamic key. After the foregoing data is obtained, the header, the raw data, and the dynamic key may be sequentially arranged, and the header, the raw data, and the dynamic key are separated by using dot separators, to form the login token.

Specifically, the header plaintext and the raw data plaintext may be encrypted by using the same Base64 algorithm, and the two obtained strings are used as the header and the raw data. Then, the header, the raw data, and the key plaintext may be encrypted by using the HMAC SHA256 algorithm, to obtain the dynamic key. Finally, the header, the raw data, and the dynamic key may be combined based on a specific sequence, to obtain the corresponding login token.

For example, the login token may be expressed as eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWliOiJ1c2VyliwiaXNzljoiI25Ly29ubmVjdCIsImV4cCI6MTU0OTg3MTM4MCwidXNlckkljoiNiJ9.pwXj8KoRbRN5iJFdNopiOTcinBIvejve9myJgKI8FC8.

The login token is returned to the client after being generated, and is stored in the flash memory of the client, so that the user can submit the login token to the serving end for verification together with an operation request when performing the next operation in the system.

An existing login token is usually generated by using a fixed key. Therefore, once a request of the client is intercepted by a hacker, it is very easy to crack or steal the login token. Consequently, security is low. In this embodiment, a token is generated based on user information of a requestor, browser information, an IP address, and the like, which increases login token generation complexity, and can prevent login from an unusual location and a cross-browser operation. In addition, user permission-related information is added to generate the login token, so that after user permission is changed, the corresponding login token becomes invalid immediately, and therefore a change of the user permission can be reflected, an unauthorized operation can be prevented, and login token security and system security can be improved.

FIG. 3 is a schematic flowchart of steps of a login token verification method according to an embodiment of this application. The method may specifically include the following steps.

S301. Receive an operation request submitted by a user by using a browser, where the operation request includes a login token issued previous time.

It should be noted that the method may be applied to various systems in which the user needs to log in before performing an operation, for example, an online banking system and a securities trading system. A scenario to which the method is specifically applied is not limited in this embodiment.

Usually, after successfully logging into a system, the user receives a valid login token issued by a serving end. The login token is usually saved in a flash memory of a client. When the user performs the next operation in the system, it may be considered that the user initiates an operation request to the serving end. The login token is included in the operation request.

The online banking system is used as an example. After the user enters a user ID and a

password to initiate a login request to the serving end, the serving end can verify the user ID and the password immediately. If the verification succeeds, the serving end allows the user to log into the online banking system, generates a login token, and returns the login token to the client. After the user logs into the system for a specific time period, if the user wants to perform an operation such as a transfer operation, the user can initiate a request to the serving end by clicking a specific button in the system. The request should carry the login token previously issued by the serving end.

For a process in which the serving end receives the login request of the user and generates the login token after the verification on the user ID and the password succeeds, refer to the detailed description in the embodiment of the login token generation method. Details are not described in this embodiment again.

S302. Extract a user ID of the user from the login token.

A login token is usually a compiled string, and includes three parts: a header, raw data, and a dynamic key. After corresponding processing is performed on the string, a user ID carried in the token can be extracted.

Therefore, after receiving the operation request submitted by the client and the login token carried in the operation request, the serving end can first identify raw data in the login token.

As described in the embodiment of the login token generation method, the header, the raw data, and the dynamic key are arranged in a specific sequence and are separated by using dot separators when being combined into the login token. Therefore, when identifying the raw data in the login token, the serving end can first determine an arrangement sequence of the raw data during generation of the login token, and extract the raw data from the login token based on the arrangement sequence.

For example, for a login token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWUiOiJ1c2VyliwiaXNzljoiib25Ly29ubmVjdCIsImV4cCl6MTU0OTg3MTM4MCwidXNlckkljoiNiJ9. pwXi8KoRbRN5iJFdNopiOTcinBIveje9myJgKI8FC8, the token is arranged in a

sequence of a header, raw data, and a dynamic key, and the parts are separated by using dot separators. Therefore, the raw data in the token can be identified as eyJzdWliOiJ1c2VyliwiaXNzljoiY25Ly29ubmVjdCIsImV4cCI6MTU0OTg3MTM4MCwidXNlckklkljoiNiJ9.

In addition, the raw data is obtained by encrypting raw data plaintext by using a specific encryption algorithm, namely, a first encryption algorithm. Therefore, after the raw data in the login token is identified, the raw data can be decrypted by using a decryption algorithm corresponding to the first encryption algorithm, to obtain the raw data plaintext.

For example, when generating the login token, the serving end encrypts the raw data plaintext by using a Base64 encryption algorithm. In this case, the serving end should perform decryption by using a Base64 decryption algorithm.

After the raw data eyJzdWliOiJ1c2VyliwiaXNzljoiY25Ly29ubmVjdCIsImV4cCI6MTU0OTg3MTM4MCwidXNlckklkljoiNiJ9 is decrypted by using the Base64 decryption algorithm, the obtained raw data plaintext is {"sub": "user", "iss": "oneconnect", "exp": "1549871380", "userID": "6"}.

Then, the user ID, namely, userID 6, of the current user can be extracted from the raw data plaintext.

S303. Obtain user information of the user corresponding to the user ID and browser information of the browser.

S304. Concatenate the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence.

S305. Generate raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypt the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token.

S306. Encrypt a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, where the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm.

S307. Combine the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token.

In this embodiment of this application, after the user ID is extracted from the login token carried in the operation request submitted by the user, the user information corresponding to the user ID and the information about the browser currently used by the user can be obtained, and the user ID, the user information, and the browser information are concatenated into the key plaintext based on the predefined concatenation method and concatenation sequence. The raw data plaintext is generated based on the user ID and the preset token expiration time, and the raw data plaintext is encrypted by using the preset first encryption algorithm, to obtain the raw data. In addition, the preset header plaintext is encrypted by using the first encryption algorithm, to obtain the header, then the header, the raw data, and the key plaintext can be encrypted by using the preset second encryption algorithm, to obtain the dynamic key, and the header, the raw data, and the dynamic key are combined into the verification token.

It should be noted that the process of generating a verification token by the serving end is similar to the process of generating a login token, and therefore is described briefly in this embodiment. For related details, refer to the description of steps S102 to S106 in the embodiment of the login token generation method. Details are not described in this embodiment again.

S308. Verify the login token by using the verification token.

After generating the verification token, the serving end can verify the received login token, to determine whether the user currently has permission to continue to perform an operation in the

system.

Specifically, the serving end can determine whether the received login token is the same as the verification token generated in real time. If the login token is the same as the verification token, it can be considered that the verification succeeds. In this case, to ensure that the user can continue to perform a subsequent operation in the system, the token expiration time of the login token can be extended, then a new login token is generated based on the extended token expiration time, and the new login token is returned to the browser. For example, the extended token expiration time can be the current time plus 10 minutes.

In addition, if the login token is different from the verification token, it can be considered that at least one of a login environment or permission of the user is changed during a time period from a time at which the user logs into the system to a current time at which the operation request is submitted. To ensure system security, the serving end can force the user to exit a current login state, and prompt the user to log in again.

In this embodiment of this application, each operation request initiated by the user in the system includes a valid login token issued previous time, so that the serving end can extract the user ID from the login token, and obtain latest user permission information, latest login/logout information, and browser information from a database based on the user ID. Then, the information can be integrated to generate a verification token to intelligently verify the currently received login token. A token validity period can be extended to generate a new token after the verification succeeds. Otherwise, the user is forced to log in again. In this embodiment, real-time verification can be performed based on latest user information and permission when an operation request is received from the user, and an intelligent comparison can be made to dynamically refresh and generate a new token, to ensure real-time validity and correctness of the token and improve system security.

FIG. 4 is a schematic diagram of a login token generation apparatus according to an embodiment of this application. The apparatus may specifically include the following modules:

a login request receiving module 401, configured to receive a login request submitted by a user by using a browser, and verify a user ID and a password that are included in the login request;

an information obtaining module 402, configured to obtain browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

a key plaintext concatenation module 403, configured to concatenate the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

a raw data encryption module 404, configured to generate raw data plaintext based on the user ID and a preset token expiration time, and encrypt the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

a dynamic key encryption module 405, configured to encrypt a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, where the header is obtained by encrypting preset header plaintext by using the first encryption algorithm; and

a login token combination module 406, configured to combine the header, the raw data, and the dynamic key into a login token.

In this embodiment of this application, the login token combination module 406 may include the following submodules:

a login token combination submodule, configured to sequentially arrange the header, the raw data, and the dynamic key, and separate the header, the raw data, and the dynamic key by using dot separators, to form the login token.

In this embodiment of this application, the browser information may include at least one of a name of the browser, version information of the browser, and current network environment information of the browser, and the network environment information may include an IP address of

a current network environment.

The user information may include permission information and login/logout information of the user, the permission information may include at least one of a user status, a user level, a user role, and the latest update time of the user role, and the login/logout information may include the latest login time and the latest logout time.

FIG. 5 is a schematic diagram of a login token verification apparatus according to an embodiment of this application. The apparatus may specifically include the following modules:

a receiving module 501, configured to receive an operation request submitted by a user by using a browser, where the operation request includes a login token issued previous time;

an extraction module 502, configured to extract a user ID of the user from the login token;

an obtaining module 503, configured to obtain user information of the user corresponding to the user ID and browser information of the browser;

a concatenation module 504, configured to concatenate the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

a first encryption module 505, configured to generate raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypt the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

a second encryption module 506, configured to encrypt a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, where the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

a combination module 507, configured to combine the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

a verification module 508, configured to verify the login token by using the verification token.

In this embodiment of this application, the extraction module 502 may specifically include the following submodules:

a raw data identification submodule, configured to identify raw data of the login token in the login token;

a raw data decryption submodule, configured to decrypt the raw data of the login token by using a decryption algorithm corresponding to the first encryption algorithm, to obtain raw data plaintext of the login token; and

a user ID extraction submodule, configured to extract the user ID of the user from the raw data plaintext of the login token.

In this embodiment of this application, the raw data identification submodule may specifically include the following units:

an arrangement sequence determining unit, configured to determine an arrangement sequence of the raw data of the login token during generation of the login token; and

a raw data extraction unit, configured to extract the raw data of the login token from the login token based on the arrangement sequence.

In this embodiment of this application, the verification module 508 may specifically include the following submodules:

a login token determining submodule, configured to determine whether the login token is the same as the verification token;

a new login token generation submodule, configured to: if the login token is the same as the

verification token, extend the token expiration time of the login token, generate a new login token based on the extended token expiration time, and return the new login token to the browser; and

a force-to-exit submodule, configured to: if the login token is different from the verification token, force the user to exit a current login state.

FIG. 6 is a schematic diagram of a server according to an embodiment of this application. As shown in FIG. 6, the server 600 in this embodiment includes a processor 610, a memory 620, and a computer-readable instruction 621 that is stored in the memory 620 and can run on the processor 610. When the processor 610 executes the computer-readable instruction 621, the steps in the embodiments of the login token generation method and the login token verification method, for example, steps S101 to S106 shown in FIG. 1 and steps S301 to S308 shown in FIG. 3, are implemented. Alternatively, when the processor 610 executes the computer-readable instruction 621, the functions of the modules/units in the foregoing apparatus embodiments, for example, the functions of the modules 401 to 406 shown in FIG. 4 and the functions of the modules 501 to 508 shown in FIG. 5, are implemented.

For example, the computer-readable instruction 621 may be divided into one or more modules/units, and the one or more modules/units are stored in the memory 620, and executed by the processor 610 to complete this application. The one or more modules/units may be instruction segments of a series of computer readable-instructions that can implement specific functions, and the instruction segments may be used to describe an execution process of the computer-readable instruction 621 in the server 600. For example, the computer-readable instruction 621 may be divided into a login request receiving module, an information obtaining module, a key plaintext concatenation module, a raw data encryption module, a dynamic key encryption module, and a login token combination module. Specific functions of the login request receiving module, the browser information and user information obtaining module, the key plaintext concatenation module, the raw data encryption module, the dynamic key encryption module, and the login token combination module are as follows:

The login request receiving module is configured to receive a login request submitted by a user by using a browser, and verify a user ID and a password that are included in the login request;

the information obtaining module is configured to obtain browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

the key plaintext concatenation module is configured to concatenate the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

the raw data encryption module is configured to generate raw data plaintext based on the user ID and a preset token expiration time, and encrypt the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

the dynamic key encryption module is configured to encrypt a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, where the header may be obtained by encrypting preset header plaintext by using the first encryption algorithm; and

the login token combination module is configured to combine the header, the raw data, and the dynamic key into a login token.

Alternatively, the computer-readable instruction 621 may be divided into a receiving module, an extraction module, an obtaining module, a concatenation module, a first encryption module, a second encryption module, a combination module, and a verification module. Specific functions of the receiving module, the extraction module, the obtaining module, the concatenation module, the first encryption module, the second encryption module, the combination module, and the verification module are as follows:

The receiving module is configured to receive an operation request submitted by a user by using a browser, where the operation request includes a login token issued previous time;

the extraction module is configured to extract a user ID of the user from the login token;

the obtaining module is configured to obtain user information of the user corresponding to the user ID and browser information of the browser;

the concatenation module is configured to concatenate the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

the first encryption module is configured to generate raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypt the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

the second encryption module is configured to encrypt a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, where the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

the combination module is configured to combine the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

the verification module is configured to verify the login token by using the verification token.

The server 600 may be a computing device such as a cloud server. The server 600 may include the processor 610, the memory 620, and the like. A person skilled in the art may understand that FIG. 6 is merely an example of the server 600 and does not constitute a limitation on the server 600. The server 600 may include more or fewer components than those shown in the figure, or may combine some components, or may include different components. For example, the server 600 may further include an input/output device, a network interface device, and a bus.

The processor 610 may be a central processing unit (CPU), or may be another general-purpose processor, a digital signal processor (DSP), an application-specific integrated circuit (ASIC), a field programmable gate array (FPGA) or another programmable logic device, a discrete gate or transistor logic device, a discrete hardware component, or the like. The general-purpose processor may be a microprocessor, or may be any conventional processor or the like.

The memory 620 may be an internal storage unit of the server 600, for example, a hard disk or memory of the server 600. Alternatively, the memory 620 may be an external storage device of the server 600, for example, a plug-connected hard disk, a smart media card (SMC), a secure digital (SD) card, or a flash card disposed on the server 600. Further, the memory 620 may include an internal storage unit of the server 600 and an external storage device. The memory 620 is configured to store the computer-readable instruction 621 and other programs and data required by the server 600. The memory 620 may be further configured to temporarily store data that has been or is to be output.

A person of ordinary skill in the art may understand that all or a part of the processes in the methods in the foregoing embodiments may be performed by a computer-readable instruction instructing related hardware. The computer-readable instruction may be stored in a nonvolatile computer-readable storage medium, and when the computer-readable instruction is executed, the processes in the embodiments of the methods may be performed. Any reference to the memory, storage, database or another medium used in the embodiments provided in this application may include a nonvolatile and/or volatile memory. The nonvolatile memory may include a read-only memory (ROM), a programmable ROM (PROM), an electrically programmable ROM (EPROM), an electrically erasable programmable ROM (EEPROM), or a flash memory. The volatile memory may include a random access memory (RAM) or an external cache memory. By way of illustration instead of limitation, the RAM can be in various forms, for example, a static RAM (SRAM), a dynamic RAM (DRAM), a synchronous DRAM (SDRAM), a double data rate SDRAM (DDRSDRAM), an enhanced SDRAM (ESDRAM), a synchlink DRAM (SLDRAM), a Rambus

direct RAM (RDRAM), a direct Rambus dynamic RAM (DRDRAM), and a Rambus dynamic RAM (RDRAM).

The foregoing descriptions are merely optional embodiments of this application, and are not intended to limit this application. For a person skilled in the art, this application may be subject to various changes and variations. Any modification, equivalent replacement, or improvement made within the spirit and principle of this application shall fall within the claims of this application.

CLAIMS

1. A login token generation method, comprising:

receiving a login request submitted by a user by using a browser, and verifying a user ID and a password that are comprised in the login request;

obtaining browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

concatenating the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext based on the user ID and a preset token expiration time, and encrypting the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

encrypting a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, wherein the header is obtained by encrypting preset header plaintext by using the first encryption algorithm; and

combining the header, the raw data, and the dynamic key into a login token.

2. The method according to claim 1, wherein the step of combining the header, the raw data, and the dynamic key into a login token comprises:

sequentially arranging the header, the raw data, and the dynamic key, and separating the header, the raw data, and the dynamic key by using dot separators, to form the login token.

3. The method according to claim 1, wherein the browser information comprises at least one of a name of the browser, version information of the browser, and current network environment information of the browser, and the network environment information comprises an IP address of a current network environment; and

the user information comprises permission information and login/logout information of the user, the permission information comprises at least one of a user status, a user level, a user role, and

the latest update time of the user role, and the login/logout information comprises the latest login time and the latest logout time.

4. A login token verification method, comprising:

receiving an operation request submitted by a user by using a browser, wherein the operation request comprises a login token issued previous time;

extracting a user ID of the user from the login token;

obtaining user information of the user corresponding to the user ID and browser information of the browser;

concatenating the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypting the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

encrypting a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, wherein the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

combining the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

verifying the login token by using the verification token.

5. The method according to claim 4, wherein the step of extracting a user ID of the user from the login token comprises:

identifying raw data of the login token in the login token;

decrypting the raw data of the login token by using a decryption algorithm corresponding to the first encryption algorithm, to obtain raw data plaintext of the login token; and

extracting the user ID of the user from the raw data plaintext of the login token.

6. The method according to claim 5, wherein the step of identifying raw data of the login token in the login token comprises:

determining an arrangement sequence of the raw data of the login token during generation of the login token; and

extracting the raw data of the login token from the login token based on the arrangement sequence.

7. The method according to claim 4, wherein the step of verifying the login token by using the verification token comprises:

determining whether the login token is the same as the verification token; and

if the login token is the same as the verification token, extending the token expiration time of the login token, generating a new login token based on the extended token expiration time, and returning the new login token to the browser; or

if the login token is different from the verification token, forcing the user to exit a current login state.

8. A login token generation apparatus, comprising:

a login request receiving module, configured to receive a login request submitted by a user by using a browser, and verify a user ID and a password that are comprised in the login request;

an information obtaining module, configured to obtain browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

a key plaintext concatenation module, configured to concatenate the user ID, the user

information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

a raw data encryption module, configured to generate raw data plaintext based on the user ID and a preset token expiration time, and encrypt the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

a dynamic key encryption module, configured to encrypt a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, wherein the header may be obtained by encrypting preset header plaintext by using the first encryption algorithm; and

a login token combination module, configured to combine the header, the raw data, and the dynamic key into a login token.

9. A login token verification apparatus, comprising:

a receiving module, configured to receive an operation request submitted by a user by using a browser, wherein the operation request comprises a login token issued previous time;

an extraction module, configured to extract a user ID of the user from the login token;

an obtaining module, configured to obtain user information of the user corresponding to the user ID and browser information of the browser;

a concatenation module, configured to concatenate the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

a first encryption module, configured to generate raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypt the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

a second encryption module, configured to encrypt a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, wherein the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

a combination module, configured to combine the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token;
and

a verification module, configured to verify the login token by using the verification token.

10. The apparatus according to claim 9, wherein the extraction module comprises:

a raw data identification submodule, configured to identify raw data of the login token in the login token;

a raw data decryption submodule, configured to decrypt the raw data of the login token by using a decryption algorithm corresponding to the first encryption algorithm, to obtain raw data plaintext of the login token; and

a user ID extraction submodule, configured to extract the user ID of the user from the raw data plaintext of the login token.

11. The apparatus according to claim 10, wherein the raw data identification submodule comprises:

an arrangement sequence determining unit, configured to determine an arrangement sequence of the raw data of the login token during generation of the login token; and

a raw data extraction unit, configured to extract the raw data of the login token from the login token based on the arrangement sequence.

12. A server, comprising a memory, a processor, and a computer-readable instruction that is stored in the memory and can run on the processor, wherein when the processor executes the computer-readable instruction, the following steps of a login token generation method are implemented:

receiving a login request submitted by a user by using a browser, and verifying a user ID and a password that are comprised in the login request;

obtaining browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

concatenating the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext based on the user ID and a preset token expiration time, and encrypting the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

encrypting a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, wherein the header is obtained by encrypting preset header plaintext by using the first encryption algorithm; and

combining the header, the raw data, and the dynamic key into a login token.

13. A server, comprising a memory, a processor, and a computer-readable instruction that is stored in the memory and can run on the processor, wherein when the processor executes the computer-readable instruction, the following steps of a login token verification method are implemented:

receiving an operation request submitted by a user by using a browser, wherein the operation request comprises a login token issued previous time;

extracting a user ID of the user from the login token;

obtaining user information of the user corresponding to the user ID and browser information of

the browser;

concatenating the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypting the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

encrypting a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, wherein the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

combining the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

verifying the login token by using the verification token.

14. The server according to claim 13, wherein when the processor executes the computer-readable instruction, the following steps of the login token verification method are further implemented:

identifying raw data of the login token in the login token;

decrypting the raw data of the login token by using a decryption algorithm corresponding to the first encryption algorithm, to obtain raw data plaintext of the login token; and

extracting the user ID of the user from the raw data plaintext of the login token.

15. The server according to claim 14, wherein when the processor executes the computer-readable instruction, the following steps of the login token verification method are further implemented:

determining an arrangement sequence of the raw data of the login token during generation of the login token; and

extracting the raw data of the login token from the login token based on the arrangement sequence.

16. A nonvolatile computer-readable storage medium, wherein the computer-readable storage medium stores a computer-readable instruction, and when the computer-readable instruction is executed by a processor, the following steps of a login token generation method are implemented:

receiving a login request submitted by a user by using a browser, and verifying a user ID and a password that are comprised in the login request;

obtaining browser information of the browser and user information of the user after the verification on the user ID and the password succeeds;

concatenating the user ID, the user information, and the browser information into key plaintext based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext based on the user ID and a preset token expiration time, and encrypting the raw data plaintext by using a preset first encryption algorithm, to obtain raw data;

encrypting a pre-generated header, the raw data, and the key plaintext by using a preset second encryption algorithm, to obtain a dynamic key, wherein the header is obtained by encrypting preset header plaintext by using the first encryption algorithm; and

combining the header, the raw data, and the dynamic key into a login token.

17. A nonvolatile computer-readable storage medium, wherein the computer-readable storage medium stores a computer-readable instruction, and when the computer-readable instruction is executed by a processor, the following steps of a login token verification method are implemented:

receiving an operation request submitted by a user by using a browser, wherein the operation

request comprises a login token issued previous time;

extracting a user ID of the user from the login token;

obtaining user information of the user corresponding to the user ID and browser information of the browser;

concatenating the user ID, the user information, and the browser information into key plaintext of a verification token based on a predefined concatenation method and concatenation sequence;

generating raw data plaintext of the verification token based on the user ID and a preset token expiration time, and encrypting the raw data plaintext of the verification token by using a preset first encryption algorithm, to obtain raw data of the verification token;

encrypting a pre-generated header of the verification token, the raw data of the verification token, and the key plaintext of the verification token by using a preset second encryption algorithm, to obtain a dynamic key of the verification token, wherein the header of the verification token is obtained by encrypting preset header plaintext of the verification token by using the first encryption algorithm;

combining the header of the verification token, the raw data of the verification token, and the dynamic key of the verification token into the verification token; and

verifying the login token by using the verification token.

18. The nonvolatile computer-readable storage medium according to claim 17, wherein when the computer-readable instruction is executed by the processor, the following steps of the login token verification method are further implemented:

identifying raw data of the login token in the login token;

decrypting the raw data of the login token by using a decryption algorithm corresponding to the first encryption algorithm, to obtain raw data plaintext of the login token; and

extracting the user ID of the user from the raw data plaintext of the login token.

19. The nonvolatile computer-readable storage medium according to claim 18, wherein when the computer-readable instruction is executed by the processor, the following steps of the login token verification method are further implemented:

determining an arrangement sequence of the raw data of the login token during generation of the login token; and

extracting the raw data of the login token from the login token based on the arrangement sequence.

20. The nonvolatile computer-readable storage medium according to claim 17, wherein when the computer-readable instruction is executed by the processor, the following steps of the login token verification method are further implemented:

determining whether the login token is the same as the verification token; and

if the login token is the same as the verification token, extending the token expiration time of the login token, generating a new login token based on the extended token expiration time, and returning the new login token to the browser; or

if the login token is different from the verification token, forcing the user to exit a current login state.