

A SECURED SYSTEM, METHOD AND COMPUTER READABLE MEDIUM FOR SYMMETRIC ENCRYPTION USING GENERATED DYNAMIC KEYS

TECHNICAL FIELD

[0001] The present disclosure relates generally to a secured system, method and computer readable medium for symmetric encryption using generated dynamic keys. Embodiments of the disclosure are related to a system, method and computer readable medium that generate dynamic keys by processing session salts and timestamps and use the key in Advanced Encryption Standard (AES) / Cipher Block Chaining (CBC) symmetric algorithm.

BACKGROUND

[0001] Symmetric encryption or symmetric key cryptography is a type of encryption scheme in which the same key (also called a symmetric key) is used for both encryption and decryption of data or messages. Symmetric encryption schemes rely on a single key that is shared between two or more users and the same key is used to encrypt and decrypt the plaintext (the data or message). The process of encryption consists of running a plaintext through an encryption algorithm called a cipher, which in turn generates a ciphertext. Consequently, the ciphertext is decryptable to plaintext by one or more users by using the key.

[0002] Advanced Encryption Standard (AES), is a symmetric block cipher to protect data. A block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key. CBC (Cipher Block Chaining) is an advanced form of block cipher encryption wherein a sequence of bits is encrypted as a single unit or block with a cipher key applied to the entire block. In CBC mode encryption, each ciphertext block is dependent on all plaintext blocks processed up to that point for its

decryption. However, AES and CBC mode encryptions are vulnerable to padding oracle attacks, which uses padding validation of a cryptographic message to decrypt the ciphertext. In cryptography, variable-length plaintext messages often have to be padded or expanded to be compatible with the underlying cryptographic primitive. Padding oracle attacks exploit the tendency of block ciphers, such as AES and CBC mode encryptions, to add arbitrary values onto the end of the last block in a sequence in order to meet the specified block size. Furthermore, at present, **128**-bit keys are not considered secure for AES and CBC mode encryptions.

[0003] In Kwourz Research LLC's, WO2018212978A1, the method providing for securely communicating traffic over a network, using a single key for encryption and decryption during a single session of communication between sender and receiver has been discussed. However, if this key is broken, the entire communication session gets compromised resulting in leakage of vital information that were meant to be kept secure. Consequently, this opens up possibilities for an unauthorized third party to figure out the pattern in the communication, and also possibly reverse engineer the logic used for encryption and decryption.

[0004] Methods of encryption and decryption provided by **BEFINE SOLUTIONS AG's** , US20170214671A1, have various shortcomings including but not limited to, dependency of both sender and receiver computing devices on a common communication server which generates encrypted data on behalf of sender and also decrypts the encrypted data on behalf of receiver. If this communication server gets compromised, then an unauthorized third party will have access to the encryption/ decryption logic and hence, will be able to access data of both the sender and the receiver. Further, since data to be encrypted is sent to the communication server for encryption, hence, if the communication channel between the sender and the communication server gets compromised, the unauthorized third party will have access to the original data as it is.

[0005] InKhoi H.'s, US20030099360A1, the method for securely transmitting a data message having a basic requirement that the parties desiring to communicate need to have knowledge of a parameter has been discussed. This would require both the parties to share the parameter over a channel which can be insecure and susceptible to attacks, where an unauthorized access of the parameter would break the entire system open. Thus, if an undesirable third party is able to access this parameter, entire communication will no longer be secure.

[0006] The symmetric encryption is relatively simple with limited length of keys and encryption strength. However, symmetric encryptions are unable to meet the present needs and requirements in cryptography. The main disadvantage of using symmetric encryption is that it is difficult and unsafe to store and exchange keys between users. Before transferring the data, a sender and a receiver must agree on the key that would be used for encryption and decryption. Once the key is chosen, both sides should remember the key. If the keys are exposed to the public, the encryption method is no longer safe. Furthermore, different senders and receivers use different keys for encryption and decryption of different data, thereby resulting in a huge number of keys that need to be stored and burdening users.

[0007] A need, therefore, exists for a secured system, method and computer readable medium for generating dynamic keys in symmetric encryption.

SUMMARY

[0008] This specification describes a system, method and computer readable medium for symmetric encryption using generated dynamic keys. Embodiments of the disclosure are related to a system, method and computer readable medium for generating dynamic keys by processing session salts and timestamps that can be used in symmetric encryption, including AES and CBC mode encryptions.

[0009] The systems, methods and computer readable media disclosed in the present application are advantageous in several ways, including but not limited to, improved security of the algorithm in encryption, dynamic keys change at every second resulting in immunity against cryptographic attacks, eliminating the need of permanently storing encryption/ decryption keys thereby making the storage of dynamic keys convenient, eliminating the need to exchange the encryption/ decryption keys and thereby removing the concerns related to exposure of the symmetric keys to an unauthorized third party, and using simple rules for processing and setting the dynamic keys for encryption and decryption.

[0010] It is therefore, one aim of the present invention to provide a system, method and computer readable medium for symmetric encryption using dynamic keys. At the client side, a session salt generator of a first computing device generates a session salt when a user logs in using a credential. A first key processor using a first set of predefined rules generates a dynamic key by processing the session salt and the timestamp of a request. The request is sent to a second computing device at the server side after encrypting the request using the dynamic key. A second key processor of the second computing device using a second rule generates a dynamic key by processing the session salt and the timestamp of the request. The request is decrypted using the dynamic key. The second computing device also generates a token based on the user information and session salt, which is sent and used by first computing device in all requests to improve the system security.

[0011] In present invention by using a session salt and timestamp as the encryption and decryption secrets makes the storage of keys simpler and easier. Using present invention, exchange of keys through different parties is no longer needed and the parties no need to worry on exposure of keys. Processing the session salt and timestamp makes the encryption/ decryption parameters change for every

second rather than predefined keys, and hence improve the security of the algorithm significantly.

[0012] In present invention, the client side will generate a random session salt as part of the dynamic key and store it in the flash memory. The session salt will remain unchanged during the same login session while the timestamp will be different from time to time. The session salt and timestamp will be put in the header while sending the request to the server side and used as the keys for encryption/decryption. Session salt and timestamp are processed with new rules before AES encryption/decryption.

[0013] The present invention provides an improved system, method and computer readable medium of cryptography based on AES and CBC mode encryptions. By using dynamic keys for the encryption and decryption, it will make the secrets of the encryption differ at every one second and there is no way to find the pattern. As the secrets are unique for each request, the encrypted data will also be different even when the original data is the same. Besides, permanent storage and exchange of keys are no longer needed since the key is only stored with each session. And there is also no concern on the exposure of keys to the public.

[0014] In a first aspect of the present disclosure, a system for symmetric encryption using at least one generated dynamic key is disclosed. The system comprising: a first computing device configured to prepare and send a request comprising a request body and a request header, the first computing device consisting: a session salt generator for generating a session salt when a user logs in using at least one credential; a memory for storing the generated session salt; a first key processor for generating a dynamic key by applying a first predefined set of rules to process the session salt and a timestamp; and a first encryption and decryption unit for encrypting a data using the dynamic key, and a second computing device configured to receive the request sent by the first computing

device, the second computing device, consisting: a database for validating the credentials of the user in the request; a second key processor for generating the dynamic key by applying the first predefined set of rules to process the sessional salt and the timestamp; and a second encryption and decryption unit for decrypting a data in the request using the dynamic key.

[0015] According to an embodiment in conjunction to the first aspect of the present disclosure, the request header comprises one or more of an Internet Protocol (IP) information, a browser information, the session salt or the timestamp.

[0016] According to an embodiment in conjunction to the first aspect of the present disclosure, the second computing device is configured to send a response and the first computing device is configured to receive the response.

[0017] According to an embodiment in conjunction to the first aspect of the present disclosure, the second computing device is further configured to retrieve at least one information about the user from the database.

[0018] According to an embodiment in conjunction to the first aspect of the present disclosure, the second computing device is further configured to prepare a secret key based on one or more information in the request header and at least one information about the user.

[0019] According to an embodiment in conjunction to the first aspect of the present disclosure, the second computing device further comprises a web token generator for generating a web token with an expiry date based on the secret key and at least one user credential and the response sent to the first computing device comprises the web token.

[0020] According to an embodiment in conjunction to the first aspect of the present disclosure, the first computing device is further configured to send the web token along with the request body and request header to the second computing device and the second computing device is configured to check the validity of the web token received from the first computing device by comparing the web token and the secret key.

[0021] According to an embodiment in conjunction to the first aspect of the present disclosure, the first set of predefined rules uses the last two digits of the timestamp to generate the dynamic key.

[0022] According to an embodiment in conjunction to the first aspect of the present disclosure, the second encryption and decryption unit is configured to encrypt the response using the second key processor before sending to the first computing device and the first encryption and decryption unit is configured to decrypt the response using the first key processor.

[0023] In a second aspect of the present disclosure, a method for symmetric encryption using at least one generated dynamic key is disclosed. The method comprising: generating, by a session salt generator of a first computing device at a client side, a session salt when a user logs in using at least one credential, storing, in a memory of the first computing device, the session salt, preparing, by the first computing device, a request header containing the session salt and a timestamp, processing, by a first key processor of the first computing device, the session salt and timestamp for generating a dynamic key by applying a first predefined set of rules, preparing, by the first computing device, a data to be encrypted, encrypting, by a first encryption and decryption unit, the data using the dynamic key, sending, by the first computing device at the client side, a request containing a request body and the request header to a second computing device

at a server side, receiving, by the second computing device, the request of the first computing device, querying, by the second computing device, the credentials of the user to a database, verifying, by a database at least one of the credentials of the user, processing, by a second key processor, the session salt and the timestamp for generating the dynamic key by applying the first predefined set of rules, decrypting, by a second encryption and decryption unit, the data in the request using the dynamic key.

[0024] According to an embodiment in conjunction to the second aspect of the present disclosure, the request header comprises one or more of an Internet Protocol (IP) information, a browser information, the session salt or the timestamp.

[0025] According to an embodiment in conjunction to the second aspect of the present disclosure, the second computing device is configured to send a response and the first computing device is configured to receive the response.

[0026] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises retrieving, by the second computing device, at least one information about the user from the database, preparing, by the second computing device, a secret key by combining at least one information about the user, and at least one information in the request header, generating, by a web token generator of the second computing device, a web token with an expiry date using the secret key and at least one user credential; and sending, by the second computing device, the response to the first computing device comprising a web token.

[0027] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises receiving, by the first computing device, the response from the second computing device containing the web token;

sending, by the first computing device, the web token, along with a subsequent request body and a subsequent request header to the second computing device; receiving, by the second computing device, the web token along with the subsequent request body and the subsequent request header; preparing, by the second computing device, a subsequent secret key by combining at least one information about the user, and at least one information in the subsequent request header; comparing, by the second computing device, the subsequent secret key with the web token to determine if the token is valid or expired; and sending, by the second computing device, a subsequent response to the first computing device.

[0028] According to an embodiment in conjunction to the second aspect of the present disclosure, the subsequent response to the first computing device comprises an error message when the token is invalid.

[0029] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises generating, by the web token generator of the second computing device, a subsequent web token with a new expiry date using the subsequent secret key and at least one user credential; and sending, by the second computing device, the subsequent response to the first computing device comprising the subsequent web token.

[0030] According to an embodiment in conjunction to the second aspect of the present disclosure, the first set of predefined rules comprises pre-processing the session salt and the timestamp to a desired character block size; obtaining a last two digits of the timestamp; assigning the last two digits of the timestamp as a first character and a second character; identifying a first pre-processed character of the session salt; determining a corresponding sequence variable of the first pre-processed character of the session salt; calculating, a move character of the first pre-processed character of the session salt such that the move character

corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp; determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the first pre-processed character of the session salt; calculating a second ASCII value of the first pre-processed character of the session salt, by adding the value of the first ASCII value and the move value; determining a character value of the second ASCII value; identifying a next first pre-processed character of the session salt; determining a corresponding sequence variable of the next first pre-processed character of the session salt; calculating, a move character of the next first pre-processed character of the session salt such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp; determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the next first pre-processed character of the session salt; calculating a second ASCII value of the next first pre-processed character of the session salt, by adding the value of the first ASCII value and the move value; and determining a character value of the second ASCII value.

[0031] According to an embodiment in conjunction to the second aspect of the present disclosure, generating the dynamic key comprises: sequentially arranging the character value of the second ASCII value of the first pre-processed session salt and the next first pre-processed session salt.

[0032] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises retrieving, by second computing device, the session salt and timestamp by applying a second set of predefined rules to the dynamic key, wherein the second set of predefined rules comprises: obtaining a last two digits of the timestamp; assigning the last two digits of the timestamp as a first character and a second character; identifying a first character

of the dynamic key; determining a corresponding sequence variable of the first character of the dynamic key; calculating, a move character of the first character of the dynamic key such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp; determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the first character of the dynamic key; calculating a second ASCII value of the first character of the dynamic key, by adding the value of the first ASCII value and the move value; determining a character value of the second ASCII value; identifying a next first character of the dynamic key; determining a corresponding sequence variable of the next first character of the dynamic key; calculating, a move character of the next first character of the dynamic key such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp; determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the next first character of the dynamic key; calculating a second ASCII value of the next first character of the dynamic key, by subtracting the move value from the value of the first ASCII value; and determining a character value of the second ASCII value; and sequentially arranging the character value of the second ASCII value of the first character of the dynamic key and the next first character of the dynamic key.

[0033] In a third aspect of the present disclosure, a computer readable medium storing a program comprising instructions that, when executed by a system, causes the system to perform the method for generating at least one dynamic key in symmetric encryption is disclosed.

[0034] The following disclosure is provided to facilitate an understanding of some of the innovative features unique to the disclosed embodiment and is not intended

to be a full description. A full appreciation of the various aspects of the embodiments disclosed herein can be gained by taking into consideration the entire specification, claims, drawings, and abstract as a whole.

BRIEF DESCRIPTION OF THE DRAWINGS

[0035] FIG. 1 is a block diagram depicting an example system **100** for symmetric encryption using generated dynamic keys, in accordance with the disclosed embodiments;

[0036] FIG. 2 is a flow diagram that illustrates an example method **200** for encrypting a data using at least one dynamic key at a first computing device **111** of the example system of FIG. 1, in accordance with the disclosed embodiments;

[0037] FIG. 3 is a flow diagram that illustrates an example method **300** for generating a web token at a second computing device **112**, of the example system of FIG. 1, in accordance with the disclosed embodiments;

[0038] FIG. 4 is a flow diagram that illustrates an example method **400** for decrypting a data using at least one dynamic key at a second computing device at a server side, in accordance with the disclosed embodiments;

[0039] FIG. 5 is a non-limiting schematic diagram that illustrates an exemplary communication **500** in an example system **100**, in accordance with the disclosed embodiments;

[0040] FIG. 6 is a flow diagram that illustrates an example method **600** for generating a dynamic key by a first key processor of a first computing device at a

client side, by applying a first set of predefined rules, in accordance with the disclosed embodiments;

[0041] FIG. 7 is a flow diagram that illustrates an example method **700** for generating a dynamic key by a second key processor of a second computing device at a server side, by applying a first set of predefined rules, in accordance with the disclosed embodiments; and

[0042] FIG. 8 is a flow diagram that illustrates an example method **800** for generating an original session salt and timestamp using a second set of predefined rules, at a server side, in accordance with the disclosed embodiments.

[0043] The summary above, as well as the following detailed description of illustrative embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the present disclosure, exemplary constructions of the disclosure are shown in the drawings. However, the disclosure is not limited to specific methods and instrumentalities disclosed herein. Moreover, those in the art will understand that the drawings are not to scale. Wherever possible, like elements have been indicated by identical numbers.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0044] The particular configurations discussed in the following description are non-limiting examples that can be varied and are cited merely to illustrate at least one embodiment and are not intended to limit the scope thereof.

[0045] FIG. 1 is a block diagram depicting an example system **100** for symmetric encryption using generated dynamic keys, in accordance with the disclosed embodiments. The system **100** mainly comprises a first computing device **111** at

a client side, a second computing device **112** at a server side and a network **110**. The first computing device **111** at the client side comprises a session salt generator **102**, a memory **104**, a first key processor **106** and a first encryption and decryption unit **108**. When a user logs in using its credentials, a random session salt is generated by the session salt generator **102** of the first computing device **111**. In some implementations of the present invention, the credentials used by the user to login may be a user identity (ID) such as an email address, mobile number, unique username or the like, and a password. 'Salt' is a unique data/ value that is used as an additional input to a one-way function, such as at the end of a password, purposed to defend against a pre-computed hash attack. 'Salting' is a process of adding such unique data/ value, that is, salt, and then hashing it to produce a different hashed value. The random session salt once generated, remains unchanged during each login session of the user. The random session salt is stored in the memory **104**. In some implementations of the present invention, the memory **104** is a flash memory which is an electronic non-volatile computer memory storage medium that can be electrically erased and reprogrammed. The first computing device **111** prepares a request header, including without limitation, an Internet Protocol (IP) information, a browser information, the session salt and a timestamp of the login request. The first key processor **106** of the first computing device **111** generates the dynamic key using the session salt generated during login and the timestamp generated for each login request. The dynamic key is generated by the first key processor **106** by applying at least one of a first set of predefined rules to the session salt and the timestamp. By processing the session salt and the timestamp, the dynamic keys generated by the first key processor **106** are different for each request. To generate the dynamic processed key, one of the first set of predefined rules uses the last two digits of the timestamp. The first encryption and decryption unit **108** encrypts the data (request body) using the generated dynamic key.

[0046] The first computing device **111** at the client side thereafter sends the request body comprising the encrypted data along with the request header,

comprising the session salt and the timestamp, to the second computing device **112** at the server side by way of the network **110**. The network **110** may be either a private network such as one within an organization which is accessible only to people within the organization; or a public network which is generally open to public access.

[0047] The second computing device **112** at the server side comprises a database **114**, a second key processor **116**, a second encryption and decryption unit **118** and a web token generator **120**. On receipt of the request (header and body) by the second computing device **112**, the second computing device **112** queries the database **114** and verifies or validates the user credentials against the values stored in the database **114**. Once the user is verified, the second computing device **112** retrieves further information about the user from the database **114**, including but not limited to user information such as user ID, user's last updated time, permission information such as user's organization status, user's organization level, all of user's role ID and role's last updated time, and user's login/logout information such as user's last login time, user's last logout time.

[0048] The second computing device **112** prepares a secret key based on the request header received from the first computing device **111**. The web token generator **120** generates a token based on the secret key. The web token is a web authentication technique that allows users to enter their credentials such as their username and password once and receiving a uniquely-generated encrypted token in exchange. This token may then be used by the user to access protected pages or resources instead of the login credentials for a designated period of time. In some implementations of the present invention, the web token generated by the server side may be a JSON Web Token (JWT). The token generated by the web token generator **120** is sent to the first computing device **111** by the second computing device **112** via the network **110**. Since, the web token is generated based on a secret key which is known only to the second computing device **112** at

the server side, if someone modifies the data accompanying the web token, the server will fail to decode it. Resultantly, the web token provides an additional layer of security in the encryption of the embodiments of the present invention. The first computing device **111** uses this token in all subsequent requests sent to the server.

[0049] The second key processor **116** of the second computing device **112** at the server side applies at least one of the first set of predefined rules to the session salt and the timestamp retrieved from the request header received from the first computing device **111**. To generate the dynamic key, one of the first set of predefined rules uses the last two digits of the timestamp. The second encryption and decryption unit **118** decrypts the data (request body) using the generated dynamic key.

[0050] It should be noted that the first and second encryption and decryption units **108** and **118** are used for both encryption and decryption purposes, depending on whether the information is sent or received. The first and second encryption and decryption units **108** and **118** perform encryption and decryption based on the predefined encryption and decryption algorithm in it. Further, the first set of predefined rules can have one or more algorithms for processing the session salt and timestamp information to generate the dynamic key.

[0051] **FIG. 2** is a flow diagram that illustrates an example method **200** for encrypting a data using at least one dynamic key at the first computing device **111** of the example system of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 2**, the example method can be implemented, for example, by logging into the first computing device by a user, as at step **202**. Then at step **204**, generating, by the session salt generator, a random session salt when the user logs in and storing, by the second computing device, the generated session salt in the memory at step **206**. The session salt remains unchanged during each login session of the user. Thereafter, as at step **208**, preparing, by the first computing

device, a request header with information, including but not limited to, an Internet Protocol (IP) information, a browser information, the session salt and a timestamp of the login request. As at steps **210** and **212**, the method **200** is implemented by obtaining, by the first key processor, the last two digits of the timestamp and processing, by the first key processor, the session salt and the timestamp using a first set of predefined rules. At step **214**, the method **300** is implemented by generating, by the first key processor, a dynamic key using processed session salt. To generate the dynamic key, one of the first set of predefined rules uses the last two digits of the timestamp. As at steps **216** and **218**, the method **200** is implemented by, preparing, by the first computing device, a data to be encrypted and encrypting, by the first computing device, the data, with at least one dynamic key thereby preparing a request body. At step **220**, sending, by the first computing device, a request comprising the encrypted data (request body) along with the request header to a second computing device by way of a network. In some implementations of the present invention, the request sent by the first computing device to the second computing device comprises a web token.

[0052] **FIG. 3** is a flow diagram that illustrates an example method **300** for generating a web token at a second computing device **112**, of the example system of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 3**, the method **300** can be implemented, for example, by receiving by the second computing device a request (header and body) of a user from a first computing device, at step **302**. The request received by the second computing device contains a request body (encrypted data) and a request header. The request header of the request contains an IP information, a browser information, a session salt and a timestamp used by the first computing device for encryption. In some implementations of the present invention, the request received by the second computing device also contains a web token, for example, a JSON web token. On receipt of the request (header and body) by the second computing device, at steps **304** and **306**, the method **300** is implemented by querying, by the second computing device to the database and verifying or validating by the second

computing device, the user credentials against the values stored in the database. At step **308**, if the user login request is verified successfully, the second computing device retrieves user credentials and information from the database, as at step **310**. The user credentials and information, include without limitation, user information such as user ID, user's last updated time, permission information such as user's organization status, user's organization level, all of user's role ID and role's last updated time, and user's login/logout information such as user's last login time, user's last logout time. If at step **308**, verification of the user is not successful, the second computing device sends a response to the first computing device, said response containing an error message, at step **326**. Thereafter, the method **300** can be implemented, for example, by generating, by the web token generator, a secret key, by combining the user's information retrieved from the database with the information received in the request header from the first computing device, such as IP address, browser information, session salt and so on, at step **312**. Then, at step **314**, forming a string with each part, that is, each information, separated by "|". For example, if a string is to be formed with the user's ID followed by the IP address, the browser information and the user's last updated date, user's last login time and so on, a secret key such as the below may be obtained:

123|198.169.6.12|Mozilla/5.0|1585125549656|1587097163179|

[0053] Thereafter, the method **300** can be implemented, for example, by checking by the second computing device, whether the received request was accompanied by a web token, for example a JSON web token at step **316**. At step **318**, if the web token is received by the second computing device, comparing, by the second computing device, the generated secret key with the web token at step **320** to determine if the token is valid or expired. If at step **322**, the web token is valid, the method **300** is implemented at step **324**, by generating, by a web token generator, a web token with a new expiry date with the generated secret key and user credentials and information. The parameters and claims used to generate the

token may vary depending on the requirement. Thereafter, the method **300** can be implemented by, sending, by the second computing device, a response to the first computing device via the network, said response containing the generated web token, at step **326**. The description of information used to build the secret key and to generate the token is non-limiting. In some implementations of the present invention, the web token generated by the server side may be a JSON Web Token (JWT).

[0054] If at step **318**, the second computing device has not received a web token from the first computing device, the web token generator will generate a web token with an expiry date, as at step **324** and thereafter the generated web token will be sent by the second computing device to the first computing device via the network, as at step **326**.

[0055] If at step **322**, the web token received from the first computing device is invalid or expired, the method **300** is implemented by, sending, by the second computing device, a response to the first computing device, said response containing an error message, at step **326**.

[0056] **FIG. 4** is a flow diagram that illustrates an example method **400** for decrypting an encrypted data using at least one dynamic key at the second computing device **112** of the example system of **FIG. 1**, in accordance with the disclosed embodiments. As disclosed in **FIG. 4**, the method **400** can be implemented, for example, by receiving by the second computing device, a request (header and body) of a user from a first computing device, at step **402**. The request received by the second computing device contains a request body (encrypted data) and a request header. The request header of the request contains an IP information, a browser information, a session salt and a timestamp used by the first computing device for encryption. In some implementations of the present invention, the request received by the second computing device also contains a

web token, for example, a JSON web token. On receipt of the request (header and body) by the second computing device, at steps **404** and **406**, the method **400** is implemented by querying, by the second computing device to the database and verifying or validating by the second computing device, the user credentials against the values stored in the database. At step **408**, if the user login request is verified successfully, the second computing device retrieves user credentials and information from the database, as at step **410**. The user credentials and information, include without limitation, user information such as user ID, user's last updated time, permission information such as user's organization status, user's organization level, all of user's role ID and role's last updated time, and user's login/logout information such as user's last login time, user's last logout time. If at step **408**, verification of the user is not successful, an error message is returned by the second computing device to the first computing device, as at step **420**.

[0057] As at steps **412** and **414**, the method **400** is implemented by obtaining, by the second key processor, the last two digits of the timestamp received in the request header from the first computing device and processing, by the second key processor, the received session salt and the timestamp using a first set of predefined rules. At step **416**, the method **400** is implemented by generating, by the second key processor, a dynamic key using the processed session salt. To generate the dynamic key, one of the first set of predefined rules uses the last two digits of the timestamp. As at step **418**, the method **400** is implemented by, decrypting, by the second computing device, the encrypted data in the request body, with at least one dynamic key. At step **420**, sending, by the second computing device, a response to the first computing device by way of a network. In some implementations of the present invention, the response sent by the second computing device to the first computing device comprises a web token.

[0058] **FIG. 5** is a non-limiting schematic diagram that illustrates an exemplary communication **500** in an example system **100**, in accordance with the disclosed

embodiments. As disclosed in **FIG. 5**, at **502**, a user logs in using credentials, at the client side. In some implementations of the present invention, the credentials used by the user to login may be a user identity (ID) such as an email address, mobile number, unique username or the like, and a password. When the user logs in, a timestamp is generated at the client side for the login request. At **504**, a random session salt is generated by the session salt generator. Then, at **506**, the random session salt is stored in the memory at the client side. Thereafter, at **508**, a request header is prepared at the client side, said header including without limitation, an Internet Protocol (IP) information, a browser information, the session salt and the timestamp of the login request. At **510**, a dynamic key is generated at the client side using the session salt generated during login and the timestamp generated for each login request. At **512**, a data is encrypted at the client side using the generated dynamic key.

[0059] Thereafter, at **514**, the client side sends a request, comprising the encrypted data (request body) and the request header, to the server side by way of the network. On receipt of the request (header and body), the server side queries the database at **516**. Then at **518**, the server side verifies or validates the user credentials against the values stored in the database. At **520**, if the user login request is verified successfully, the server side obtains the last two digits of the timestamp received in the request header from the client side and at **522**, the server side processes the received session salt and the timestamp using a first set of predefined rules. At **524**, a dynamic key is generated at the server side using the processed session salt. Thereafter, at **526**, the received encrypted data or response body is decrypted at the server side with at least one dynamic key. At **528**, the server side sends a response to the client side by way of the network.

[0060] **FIG. 6** is a flow diagram that illustrates an example method **600** for generating a dynamic key by a first key processor of a first computing device at a client side, by applying a first set of predefined rules, in accordance with the

disclosed embodiments. As disclosed in **FIG. 6**, the first set of predefined rules can be applied **600**, for example, at step **602**, by pre-processing a session salt and a timestamp generated at the first computing device to a desired character block size. The pre-processing may include steps such as padding the characters of the session salt and the timestamp to reach the desired character block size. In an example, if the desired character block size is **16**, and the length of the session salt or the timestamp or both is less than **16**, the same are padded to reach a **16** character block size by adding “0” at the beginning of the characters till the desired character block size is reached. In the above example, if the timestamp originally is “**1577946842573**”, after padding, it will become “**0001577946842573**” to reach the desired character block size of **16**. In some implementations, if the length of the session salt or the timestamp or both is more than the desired character block size, only the requisite characters as per the desired character block size are considered and the remaining are disregarded. In the above example, if the session salt is “**835ad5d0808711ea8f687ba26b764bfb**”, and the desired character block size is **16**, only “**835ad5d0808711ea**” will be used for the subsequent steps in the application **600**.

[0061] Thereafter, the first set of predefined rules can be applied **600**, as at step **604** and **606**, by obtaining the last two digits of the padded timestamp and assigning these two digits to a variable “mod” and a variable “k”. In some implementations of the present invention, the “mod” and “k” refers to the first and second character of the last two digits of the timestamp, respectively. For example, if the timestamp after padding is “**0001577946842573**”, then the last two digits will be **7** and **3**; and accordingly, the value of “mod” and “k” will be **7** and **3**.

[0062] Thereafter, as disclosed in **FIG. 6**, the method **600** can be implemented, at step **608**, a first pre-processed character of the session salt is identified and at step **610**, a corresponding sequence variable, “i”, of the first pre-processed character in the character block of the session salt is determined, such that the “i” value for the

first character is “0”. For example, if the pre-processed session salt is “835ad5d0808711ea”, and if the first pre-processed character is “8”, then the corresponding sequence “i” is “0”. At step **612**, a move for the character is calculated using a formula. The move here refers to the move calculated at the client side and also refers as a first move. In some implementations of the present invention, the ‘move’ is calculated using the formula “ $k+i\%mod$ ”, wherein “i” represents the sequence number of the character such that for the first character, the value of ‘i’ is 0, and ‘%’ represents a modulo operation which is a mathematical operation, like a division operator, except that the answer is the remainder of an integer division operation, rather than a decimal result. In the above example, if the pre-processed session salt and timestamp are “835ad5d0808711ea” and “0001577946842573”, respectively, and if the first pre-processed session salt is “8”, the “i” is “0” and the value of the corresponding “move” for said character is “3”.

[0063] Then, at step **614**, the value of the first pre-processed character of the session salt is converted to a first ASCII (American Standard Code for Information Interchange) value. ASCII is a character encoding standard for electronic communication. ASCII encodes **128** specified characters (alphabetic, numeric, or special character) into 7-bit integers. In the above example, if the first pre-processed character is “8”, the first ASCII value for said character would be “56”. Thereafter, at step **616**, a second ASCII value is calculated by adding the first ASCII value and the corresponding “move” value of the first pre-processed character of the session salt. In the above example, if the “move” value is “3” and the first ASCII value is “56”, then at step **616**, the second ASCII value is “59”.

[0064] Thereafter, as disclosed in **FIG. 6**, the method **600** can be implemented, at step **618**, the second ASCII value is converted to a corresponding character format using ASCII conversion to obtain a character of a processed session salt. At step **620**, the first key processor checks if a pre-processed character is remaining for

processing. At step **622**, if a pre-processed character is remaining for processing, a next first pre-processed character of the session salt is identified at step **608**, and thereafter the method steps **610**, **612**, **614**, **616**, **618** and **620** are implemented to process the next first pre-processed character. As disclosed in **FIG. 6**, the method **600** can be applied, for example, by performing the above steps for all or as many characters and character blocks as desired.

[0065] At step **622**, if all characters of the pre-processed session salt have been processed, at step **624**, the first key processor generates the dynamic key by sequentially arranging the character formats of the processed session salt obtained at step **618**. In the above example, if the pre-processed session salt and timestamp are “**835ad5d0808711ea**” and “**0001577946842573**”, respectively, the processed session salt is “**;7:gk=m3<5>>9:he**”, which is the dynamic key.

[0066] Thereafter, at step **626**, the method **600** is implemented by encrypting the data using a symmetric encryption method, with the dynamic key before sending it to a second computing device. In some implementations of the present invention, the method used for encrypting the request body may be AES encryption where the dynamic key would be used for encryption.

[0067] The above example is explained in consolidation below wherein the method **600** has been implemented. For example, if the random session salt and timestamp generated by the first computing device are “**835ad5d0808711ea8f687ba26b764bfb**” and “**0001577946842573**”, and the desired character block size is “**16**”, the pre-processed session salt and timestamp is “**835ad5d0808711ea**” and “**0001577946842573**”, respectively. The values of “**k**” and “**mod**” are **3** and **7**, respectively. The below “**Table 1**” illustrates the values and its calculation for parameters, such as the first ASCII value for each pre-processed character of the session salt, corresponding move value, the second ASCII value and the corresponding characters of the processed session salt. The session salt

after processing is obtained by sequentially arranging the obtained characters, which is “;7:gk=m3<5>>9:he” in the above example. Accordingly, “;7:gk=m3<5>>9:he” is the dynamic key used for encryption of data by the first computing device. It should be noted that the data encryption and decryption is performed using the dynamic key obtained after processing session salt and timestamp. Each request body will be encrypted using the dynamic key generated at the client side before sending to the server side. Before encrypting the data, for example, in AES encryption, the session salt and timestamp are processed using the first set of predefined rules, and therefore, the dynamic key for encryption is different for each request sent by the first computing device to the second computing device.

TABLE 1

Old Characters	Old ASCII value	Sequence “i”	Move (k+i%mode)	New ASCII value	New Characters
8	56	0	3	59	;
3	51	1	4	55	7
5	53	2	5	58	:
a	97	3	6	103	g
d	100	4	7	107	k
5	53	5	8	61	=

Old Characters	Old ASCII value	Sequence "i"	Move (k+i%mode)	New ASCII value	New Characters
d	100	6	9	109	m
0	48	7	3	51	3
8	56	8	4	60	<
0	48	9	5	53	5
8	56	10	6	62	>
7	55	11	7	62	>
1	49	12	8	57	9
1	49	13	9	58	:
e	101	14	3	104	h
a	97	15	4	101	e

[0068] FIG. 7 is a flow diagram that illustrates an example method 700 for generating a dynamic key by a second key processor of a second computing device at a server side, by applying a first set of predefined rules, in accordance with the disclosed embodiments. As disclosed in FIG. 7, the first set of predefined

rules can be applied **700**, for example, at step **702**, by pre-processing a session salt and a timestamp received in a request header by the second computing device from a first communication device, to a desired character block size. The pre-processing may include steps such as padding the characters of the session salt and the timestamp to reach the desired character block size. In an example, if the desired character block size is 16, and the length of the session salt or the timestamp or both is less than 16, the same are padded to reach a 16 character block size by adding “0” at the beginning of the characters till the desired character block size is reached. In the above example, if the timestamp originally is “1577946842573”, after padding, it will become “0001577946842573” to reach the desired character block size of 16. In some implementations, if the length of the session salt or the timestamp or both is more than the desired character block size, only the requisite characters as per the desired character block size are considered and the remaining are disregarded. In the above example, if the session salt is “835ad5d0808711ea8f687ba26b764bfb”, and the desired character block size is **16**, only “**835ad5d0808711ea**” will be used for the subsequent steps in the application **700**.

[0069] Thereafter, the first set of predefined rules can be applied **700**, as at step **704** and **706**, by obtaining the last two digits of the padded timestamp and assigning these two digits to a variable “mod” and a variable “k”. In some implementations of the present invention, the “mod” and “k” refers to the first and second character of the last two digits of the timestamp, respectively. For example, if the timestamp after padding is “0001577946842573”, then the last two digits will be 7 and 3; and accordingly, the value of “mod” and “k” will be 7 and 3.

[0070] Thereafter, as disclosed in **FIG. 7**, the method **700** can be implemented, at step **708**, a first pre-processed character of the session salt is identified and at step **710**, a corresponding sequence variable, “i”, of the first pre-processed character in the character block of the session salt is determined, such that the “i” value for the

first character is “0”. For example, if the pre-processed session salt is “835ad5d0808711ea”, and if the first pre-processed character is “8”, then the corresponding sequence “i” is “0”. At step **712**, a move for the character is calculated using a formula. The move here refers to the move calculated at the server side and also refers as a first move. In some implementations of the present invention, the ‘move’ is calculated using the formula “ $k+i\%mod$ ”, wherein “i” represents the sequence number of the character such that for the first character, the value of ‘i’ is 0, and ‘%’ represents a modulo operation which is a mathematical operation, like a division operator, except that the answer is the remainder of an integer division operation, rather than a decimal result. In the above example, if the pre-processed session salt and timestamp are “835ad5d0808711ea” and “0001577946842573”, respectively, and if the first pre-processed session salt is “8”, the “i” is “0” and the value of the corresponding “move” for said character is “3”.

[0071] Then, at step **714**, the value of the first pre-processed character of the session salt is converted to a first ASCII (American Standard Code for Information Interchange) value. ASCII is a character encoding standard for electronic communication. ASCII encodes 128 specified characters (alphabetic, numeric, or special character) into 7-bit integers. In the above example, if the first pre-processed character is “8”, the first ASCII value for said character would be “56”. Thereafter, at step **716**, a second ASCII value is calculated by adding the first ASCII value and the corresponding “move” value of the first pre-processed character of the session salt. In the above example, if the “move” value is “3” and the first ASCII value is “56”, then at step 716, the second ASCII value is “59”.

[0072] Thereafter, as disclosed in **FIG. 7**, the method **700** can be implemented, at step **718**, the second ASCII value is converted to a corresponding character format using ASCII conversion to obtain a character of a processed session salt. At step **720**, the second key processor checks if a pre-processed character is remaining

for processing. At step **722**, if a pre-processed character is remaining for processing, a next first pre-processed character of the session salt is identified at step **708**, and thereafter the method steps **710**, **712**, **714**, **716**, **718** and **720** are implemented to process the next first pre-processed character. As disclosed in **FIG. 7**, the method **700** can be applied, for example, by performing the above steps for all or as many characters and character blocks as desired.

[0073] At step **722**, if all characters of the pre-processed session salt have been processed, at step **724**, the second key processor generates the dynamic key by sequentially arranging the character formats of the processed session salt obtained at step **718**. In the above example, if the pre-processed session salt and timestamp are “835ad5d0808711ea” and “0001577946842573”, respectively, the processed session salt is “;7:gk=m3<5>>9:he”, which is the dynamic key.

[0074] Thereafter, at step **726**, the method **700** is implemented by decrypting the received encrypted data (response body) using a symmetric encryption method, with the dynamic key. In some implementations of the present invention, the method used for decrypting the request body may be AES encryption/ decryption where the dynamic key would be used for decryption.

[0075] **FIG. 8** is a flow diagram that illustrates an example method **800** for generating an original session salt and timestamp using a second set of predefined rules, at a server side, in accordance with the disclosed embodiments. The original session salt and timestamp refers to the session salt and timestamp generated at a client side and thereafter processed to encrypt a data at the client side which is sent to the server side for decryption. The example method **800** starts at step **802** and checks if the original session salt and timestamp are required to be retrieved/calculated. If at step **804**, original session salt and timestamp are not required, the method **800** terminates at step **806**. However, if at step **804**, the original session salt and timestamp are required to be calculated back from the

dynamic key (processed session salt) and the timestamp, the method **800** is implemented, at step **808**, by obtaining the last two digits of the timestamp and, at step **810**, assigning these two digits to a variable “mod” and a variable “k”. In some implementations of the present invention, the “mod” and “k” refers to the first and second character of the last two digits of the timestamp, respectively. For example, if the timestamp is “0001577946842573”, then the last two digits will be 7 and 3; and accordingly, the value of “mod” and “k” will be 7 and 3.

[0076] Thereafter, as disclosed in **FIG. 8**, the method **800** can be implemented, at step **812**, by identifying a first character of the processed session salt (dynamic key) and at step **814**, by determining a corresponding sequence variable, “i”, of the first character of the processed session salt, such that the “i” value for the first character is “0”. For example, if the processed session salt is “;7:gk=m3<5>>9:he”, and if the first character is “;”, then the corresponding sequence “i” is “0”. At step **816**, a move for the character is calculated using a formula. The move here refers to the move calculated at the server side and also refers as a second move. In some implementations of the present invention, the ‘move’ is calculated using the formula “ $k+i\%mod$ ”, wherein “i” represents the sequence number of the character such that for the first character of the processed session salt, the value of ‘i’ is 0, and ‘%’ represents a modulo operation which is a mathematical operation, like a division operator, except that the answer is the remainder of an integer division operation, rather than a decimal result. In the above example, if the processed session salt and timestamp are “;7:gk=m3<5>>9:he” and “0001577946842573”, respectively, and if the first session salt is “;”, the “i” is “0” and the value of the corresponding “move” for said character is “3”.

[0077] Then, at step **818**, the value of the first character of the processed session salt is converted to a first ASCII (American Standard Code for Information Interchange) value. ASCII is a character encoding standard for electronic communication. ASCII encodes 128 specified characters (alphabetic, numeric, or

special character) into 7-bit integers. In the above example, if the first character is “,”, the first ASCII value for said character would be “59”. Thereafter, at step **820**, a second ASCII value is calculated by subtracting from the first ASCII value, the corresponding “move” value of the first character of the processed session salt. In the above example, if the “move” value is “3” and the first ASCII value is “59”, then at step **820**, the second ASCII value is “56”.

[0078] Thereafter, as disclosed in **FIG. 8**, the method **800** can be implemented, at step **822**, the second ASCII value is converted to a corresponding character format using ASCII conversion to obtain a character of a processed session salt. At step **824**, the second key processor checks if a character is remaining for processing. At step **826**, if a character is remaining for processing, a next first character of the session salt is identified at step **812**, and thereafter the method steps **814**, **816**, **818**, **820**, **822** and **824** are implemented to process the next first character.

[0079] At step **826**, if all characters of the session salt have been processed, at step **828**, the second key processor generates the original session salt by sequentially arranging the character formats obtained at step **822**. In the above example, if the session salt and timestamp are “;7:gk=m3<5>>9:he” and “0001577946842573”, respectively, the original session salt that is generated from the second set of predefined rules is “835ad5d0808711ea”, which is the original session salt used for encryption at the client side.

[0080] The present invention provides a system, method and computer readable medium for enabling cryptography based on AES and CBC mode encryptions. By using dynamic keys for the encryption and decryption, the secrets of the encryption change at every second, therefore leaving no way for a third party to find the pattern in the encrypted communication. Further, since the keys are unique for each request, the encrypted data can also be different even when the original data is the same. Furthermore, the present invention has also eliminated the need for

permanent storage and exchange of keys are no longer needed since each key is only stored for a single session.

[0081] The systems, methods and computer readable media disclosed in the present application are advantageous in several ways, including but not limited to, improved security of the algorithm in encryption, dynamic keys change at every second resulting in immunity against cryptographic attacks, eliminating the need of permanently storing encryption/decryption keys thereby making the storage of dynamic keys convenient, eliminating the need to exchange the encryption/decryption keys and thereby removing the concerns related to exposure of the symmetric keys to an unauthorized third party, and using simple rules for processing and setting the dynamic keys for encryption and decryption.

[0082] It will be appreciated that variations of the above disclosed and other features and functions, or alternatives thereof, may be desirably combined into many other different systems or applications. Also, various presently unforeseen or unanticipated alternatives, modifications, variations or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

[0083] Although embodiments of the current disclosure have been described comprehensively in considerable detail to cover the possible aspects, those skilled in the art would recognize that other versions of the disclosure are also possible.

CLAIMS

What is claimed is:

- 1.** A system for symmetric encryption using at least one generated dynamic key, comprising:

 - a first computing device configured to prepare and send a request comprising a request body and a request header, the first computing device consisting:
 - a session salt generator for generating a session salt when a user logs in using at least one credential;
 - a memory for storing the generated session salt;
 - a first key processor for generating a dynamic key by applying a first predefined set of rules to process the sessional salt and a timestamp; and
 - a first encryption and decryption unit for encrypting a data using the dynamic key, and a second computing device configured to receive the request sent by the first computing device, the second computing device, consisting:
 - a database for validating the credentials of the user in the request;
 - a second key processor for generating the dynamic key by applying the first predefined set of rules to process the sessional salt and the timestamp; and
 - a second encryption and decryption unit for decrypting a data in the request using the dynamic key.
- 2.** The system of claim **1**, wherein the request header comprises one or more of an Internet Protocol (IP) information, a browser information, the session salt or the timestamp.
- 3.** The system of claim **1** or **2**, wherein the second computing device is configured to send a response and the first computing device is configured to receive the response.

4. The system of claim **3**, wherein the second computing device is further configured to retrieve at least one information about the user from the database.

5. The system of claim **4**, wherein the second computing device is further configured to prepare a secret key based on one or more information in the request header and at least one information about the user.

6. The system of claim **5**, wherein the second computing device further comprises a web token generator for generating a web token with an expiry date based on the secret key and at least one user credential and the response sent to the first computing device comprises the web token.

7. The system of claim **6**, wherein the first computing device is further configured to send the web token along with the request body and request header to the second computing device and the second computing device is configured to check the validity of the web token received from the first computing device by comparing the web token and the secret key.

8. The system as claimed in claim **1**, wherein the first set of predefined rules uses the last two digits of the timestamp to generate the dynamic key.

9. The system of claim **3**, wherein the second encryption and decryption unit is configured to encrypt the response using the second key processor before sending to the first computing device and the first encryption and decryption unit is configured to decrypt the response using the first key processor.

10. A method for symmetric encryption using at least one generated dynamic key, comprising:

generating, by a session salt generator of a first computing device at a client side, a session salt when a user logs in using at least one credential,

storing, in a memory of the first computing device, the session salt,
preparing, by the first computing device, a request header containing the session salt and a timestamp,
processing, by a first key processor of the first computing device, the session salt and timestamp for generating a dynamic key by applying a first predefined set of rules,
preparing, by the first computing device, a data to be encrypted,
encrypting, by a first encryption and decryption unit, the data using the dynamic key,
sending, by the first computing device at the client side, a request containing a request body and the request header to a second computing device at a server side,
receiving, by the second computing device, the request of the first computing device,
querying, by the second computing device, the credentials of the user to a database,
verifying, by a database at least one of the credentials of the user,
processing, by a second key processor, the session salt and the timestamp for generating the dynamic key by applying the first predefined set of rules, decrypting, by a second encryption and decryption unit, the data in the request using the dynamic key.

11. The method of claim **10**, wherein the request header comprises one or more of an Internet Protocol (IP) information, a browser information, the session salt or the timestamp.

12. The method of claim **10** or **11**, wherein the second computing device is configured to send a response and the first computing device is configured to receive the response.

13. The method of claim **12**, further comprising:

retrieving, by the second computing device, at least one information about the user from the database,
preparing, by the second computing device, a secret key by combining at least one information about the user, and at least one information in the request header,
generating, by a web token generator of the second computing device, a web token with an expiry date using the secret key and at least one user credential; and
sending, by the second computing device, the response to the first computing device comprising the web token.

14. The method of claim **13**, further comprising:

receiving, by the first computing device, the response from the second computing device containing the web token;

sending, by the first computing device, the web token, along with a subsequent request body and a subsequent request header to the second computing device;

receiving, by the second computing device, the web token along with the subsequent request body and the subsequent request header;

preparing, by the second computing device, a subsequent secret key by combining at least one information about the user, and at least one information in the subsequent request header;

comparing, by the second computing device, the subsequent secret key with the web token to determine if the token is valid or expired; and

sending, by the second computing device, a subsequent response to the first computing device.

15. The method of claim **14**, wherein the subsequent response to the first computing device comprises an error message when the token is invalid.

16. The method of claim **14**, further comprising:

generating, by the web token generator of the second computing device, a subsequent web token with a new expiry date using the subsequent secret key and at least one user credential; and
 sending, by the second computing device, the subsequent response to the first computing device comprising the subsequent web token.

17. The method of claim **10**, wherein the first set of predefined rules comprises:
 pre-processing the session salt and the timestamp to a desired character block size;
 obtaining a last two digits of the timestamp;
 assigning the last two digits of the timestamp as a first character and a second character;
 identifying a first pre-processed character of the session salt;
 determining a corresponding sequence variable of the first pre-processed character of the session salt;
 calculating, a move character of the first pre-processed character of the session salt such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp;
 determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the first pre-processed character of the session salt;
 calculating a second ASCII value of the first pre-processed character of the session salt, by adding the value of the first ASCII value and the move value;
 determining a character value of the second ASCII value;
 identifying a next first pre-processed character of the session salt;
 determining a corresponding sequence variable of the next first pre-processed character of the session salt;
 calculating, a move character of the next first pre-processed character of the session salt such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp;

determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the next first pre-processed character of the session salt; calculating a second ASCII value of the next first pre-processed character of the session salt, by adding the value of the first ASCII value and the move value; and determining a character value of the second ASCII value.

18. The method of claim **17**, wherein generating the dynamic key comprises: sequentially arranging the character value of the second ASCII value of the first pre-processed session salt and the next first pre-processed session salt.

19. The method of claim **10**, further comprising retrieving, by second computing device, the session salt and timestamp by applying a second set of predefined rules to the dynamic key, wherein the second set of predefined rules comprises: obtaining a last two digits of the timestamp; assigning the last two digits of the timestamp as a first character and a second character; identifying a first character of the dynamic key; determining a corresponding sequence variable of the first character of the dynamic key; calculating, a move character of the first character of the dynamic key such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp; determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the first character of the dynamic key; calculating a second ASCII value of the first character of the dynamic key, by adding the value of the first ASCII value and the move value; determining a character value of the second ASCII value; identifying a next first character of the dynamic key; determining a corresponding sequence variable of the next first character of the dynamic key;

calculating, a move character of the next first character of the dynamic key such that the move character corresponds to the second character of last two characters of the timestamp added to percentage MOD function value of the sequence variable and the first character of the time stamp;
determining, a first American Standard Code for Information Interchange (ASCII) value corresponding to the next first character of the dynamic key;
calculating a second ASCII value of the next first character of the dynamic key, by subtracting the move value from the value of the first ASCII value; and
determining a character value of the second ASCII value; and
sequentially arranging the character value of the second ASCII value of the first character of the dynamic key and the next first character of the dynamic key.

20. A computer readable medium storing a program comprising instructions that, when executed by a system, cause the system to perform the method of claims **10** to **19**.