(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 July 2014 (03.07.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/106149 A1

(51) International Patent Classification:
G06F 21/00 (2013.01) *H04L 9/30* (2006.01)
H04L 9/00 (2006.01)

(21) International Application Number:
PCT/US2013/078213

(22) International Filing Date:
30 December 2013 (30.12.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/747,478 31 December 2012 (31.12.2012) US

(71) Applicant: **SAFELYLOCKED, LLC** [US/US]; 1651 N. Pelham Rd., NE, Suite A, Atlanta, GA 30324 (US).

(72) Inventor: **HURSTI, Harri**; 1651 N. Pelham Rd., NE, Suite A, Atlanta, GA 30324 (US).

(74) Agent: **LYON, John, L.**; Thomas Horstemeyer, LLP, 400 Interstate North Parkway, Suite 1500, Atlanta, GA 30339 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

[Continued on next page]

(54) Title: TECHNIQUES FOR VALIDATING CRYPTOGRAPHIC APPLICATIONS

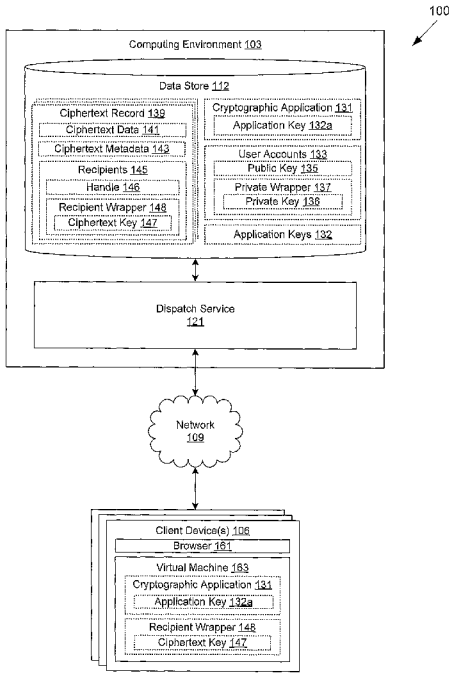

FIG. 1

(57) Abstract: Disclosed are various embodiments for validating cryptographic applications. A cryptographic application is transmitted to a client device. Subsequently, a communication link is established with the transmitted cryptographic application as it executes in the client device. A round-trip time for communications with the transmitted cryptographic application is measured. Validation data and expected response data is generated, and the validation data is sent to the previously transmitted cryptographic application as it executes in the client device.

WO 2014/106149 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

TECHNIQUES FOR VALIDATING CRYPTOGRAPHIC APPLICATIONS

CLAIM OF PRIORITY

[0001] This Application claims priority to U.S. Provisional Application Serial No. 61/747,478, filed on December 31, 2012, which is incorporated by reference in its entirety as if fully set forth herein.

BACKGROUND

[0002] In an age of information, people can produce substantial quantities of data. Those originating the data can wish to keep some of the data confidential as to the general public, while also sharing the data with select recipients. Traditional data security architectures suffer from vulnerabilities that can compromise the confidence of the data as it is stored and as it traverses networks such as the Internet.

SUMMARY

[0003] Disclosed are various techniques for the secure encryption, storage, distribution, and decryption of data for an end-user. According to various embodiments, a cryptographic application can be obtained for validation and execution in a client device. The cryptographic application can offer various services, including encrypting plaintext data available to the client device. In response to an encryption request, the cryptographic application can generate a ciphertext key with which to configure the encryption algorithm. The cryptographic application implementing the encryption algorithm can produce ciphertext data as output based upon the plaintext data as input. The cryptographic application can also encrypt the ciphertext key within a recipient wrapper, where an encryption algorithm is configured with a recipient key that can be unique for each recipient.

The ciphertext data and the recipient wrapper can then be transmitted via a network to one or more remote computing devices for later retrieval.

[0004] In an embodiment, a system is provided comprising: a computing device; and an application executable in the computing device, the application comprising: logic that transmits a cryptographic application to a client device; logic that establishes a communication link with the transmitted cryptographic application, wherein the transmitted cryptographic application is executing in the client device subsequent to transmission; logic that measures a round-trip time for a communication with the transmitted cryptographic application executing in the client device; logic that generates validation data and expected response data, wherein the validation data and the expected response data are based at least in part on a cryptographic key embedded in the transmitted cryptographic application executing in in the client device; and logic that sends the validation data to the transmitted cryptographic application executing in the client device. In any one or more embodiments, the application can comprise logic that detects whether a response is received from the transmitted cryptographic application executing in the client device within a time window comprising the measured round-trip time plus a delay tolerance. In any one or more embodiments, the application can comprise logic that retransmits the cryptographic application to the client device in response to detecting that the response is not received. In any one or more embodiments, the application can comprise logic that retransmits the cryptographic application to the client device in response to detecting that the response is received outside of the time window. In any one or more embodiments, the application can comprise logic that determines whether the response is valid based at least in part on a comparison of the response with the expected response data; and logic that logs whether the response is valid.

In any one or more embodiments, the application can comprise logic that sends a termination message to the transmitted cryptographic application executing in the client device in response to a determination that the response is invalid. In any one or more embodiments, the application can comprise logic that generates an application key; logic that embeds the application key within the cryptographic application; and logic that obfuscates at least a portion of the cryptographic application.

[0005] In an embodiment, a method is provided comprising: transmitting, via a computing device, a cryptographic application to a client device; establishing, via the computing device, a communication link with the transmitted cryptographic application, wherein the transmitted cryptographic application is executing in the client device subsequent to transmission; generating, via the computing device, validation data and expected response data, wherein the validation data and the expected response data are based at least in part on a cryptographic key embedded in the transmitted cryptographic application executing in in the client device; and sending, via the computing device, the validation data to the transmitted cryptographic application executing in the client device. In any one or more embodiments, the method can comprise measuring, via the computing device, a round-trip time for a communication with the transmitted cryptographic application executing in the client device, and detecting, via the computing device, whether a response is received from the transmitted cryptographic application executing in the client device within a time window comprising the measured round-trip time plus a delay tolerance. In any one or more embodiments, the method can comprise retransmitting, via the computing device, the cryptographic application to the client device in response to detecting that the response is not received. In any one or

more embodiments, the method can comprise retransmitting, via the computing device, the cryptographic application to the client device in response to detecting that the response is received outside of the time window. In any one or more embodiments, the method can comprise determining, via the computing device, whether the response is valid based at least in part on a comparison of the response with the expected response data; and logging, via the computing device, a result of determining whether the response is valid. In any one or more embodiments, the method can comprise transmitting, via the computing device, a termination message to the transmitted cryptographic application executing in the client device in response to determining that the response is invalid. In any one or more embodiments, the method can comprise generating, via the computing device, an application key; embedding, via the computing device, the application key within the cryptographic application; and obfuscating, via the computing device, at least a portion of the cryptographic application.

[0006] In an embodiment, a non-transitory computer readable medium including a program is provided, the program comprising: code that transmits a cryptographic application to a client device; code that establishes a communication link with the transmitted cryptographic application, wherein the transmitted cryptographic application is executing in the client device subsequent to transmission; code that generates validation data and expected response data, wherein the validation data and the expected response data are based at least in part on a cryptographic key embedded in the transmitted cryptographic application executing in in the client device; and code that sends the validation data to the transmitted cryptographic application executing in the client device. In one or more embodiments, the program can comprise code that measures a round-trip time for a communication with the

transmitted cryptographic application executing in the client device, and code that that detects whether a response is received from the transmitted cryptographic application executing in the client device within a time window comprising the measured round-trip time plus a delay tolerance. In one or more embodiments, the program can comprise code that retransmits the cryptographic application to the client device in response to detecting that the response is not received. In one or more embodiments, the program can comprise code that retransmits the cryptographic application to the client device in response to detecting that the response is received outside of the time window. In one or more embodiments, the program can comprise code that determines that the response is valid based at least in part on a comparison of the response with the expected response data; and code that logs whether the response is valid. In one or more embodiments, the program can comprise code that sends a termination message to the transmitted cryptographic application executing in the client device in response to a determination that the response is invalid.

[0007] Other systems, methods, features, and advantages of the present disclosure for the secure encryption storage, distribution and decryption of data of an end user will be or become apparent to one with skill in the art upon examination of the following drawings and detailed description. It is intended that all such additional systems, methods, features, and advantages be included within this description, be within the scope of the present disclosure, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Many aspects of the present disclosure can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, with emphasis instead being placed upon clearly illustrating the principles of the disclosure. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

[0009] FIG. 1 is a drawing of a networked environment according to various embodiments of the present disclosure.

[0010] FIG. 2 is a flowchart illustrating one example of functionality implemented as portions of a dispatch service executed in a computing environment in the networked environment of FIG. 1 according to various embodiments of the present disclosure.

[0011] FIG. 3 is a flowchart illustrating one example of functionality implemented as portions of a dispatch service executed in the computing environment in the networked environment of FIG. 1 according to various embodiments of the present disclosure.

[0012] FIG. 4 is a schematic block diagram that provides one example illustration of a client device employed in the networked environment of FIG. 1 according to various embodiments of the present disclosure.

DETAILED DESCRIPTION

[0013] Disclosed are various techniques for the secure encryption, storage, distribution, and decryption of data for an end-user. According to various embodiments, a cryptographic application can be obtained for validation and

execution in a client device. The cryptographic application can offer various services, including encrypting plaintext data available to the client device. In response to an encryption request, the cryptographic application can generate a ciphertext key with which to configure the encryption algorithm. The cryptographic application implementing the encryption algorithm can produce ciphertext data as output based upon the plaintext data as input. The cryptographic application can also encrypt the ciphertext key within a recipient wrapper, where an encryption algorithm is configured with a recipient key that can be unique for each recipient. The ciphertext data and the recipient wrapper can then be transmitted via a network to one or more remote computing devices for later retrieval.

[0014] A recipient can access the particular ciphertext data shared with him or her through use of an identifier, such as a uniform resource identifier (URI), which can initiate on-demand retrieval and/or execution of the cryptographic application in the client device. The cryptographic application can retrieve the ciphertext data and the recipient wrapper from the remote computing device(s). The recipient can apply the appropriate key to the cryptographic application in order to decrypt the ciphertext key from the recipient wrapper. Thereafter, the cryptographic application can decrypt the ciphertext data using the ciphertext key. In the following discussion, a general description of the system and its components is provided, followed by a discussion of the operation of the same.

[0015] With reference to FIG. 1, shown is a networked environment 100 according to various embodiments. The networked environment 100 includes a computing environment 103 and a client device 106, which are in data communication with each other via a network 109. The network 109 includes, for example, the Internet, intranets, extranets, wide area networks (WANs), local area

networks (LANs), wired networks, wireless networks, or other suitable networks, *etc.*, or any combination of two or more such networks.

[0016] The computing environment 103 can comprise, for example, a server computer or any other system providing computing capability. Alternatively, the computing environment 103 can employ a plurality of computing devices that can be employed that are arranged, for example, in one or more server banks or computer banks or other arrangements. Such computing devices can be located in a single installation or can be distributed among many different geographical locations. For example, the computing environment 103 can include a plurality of computing devices that together can comprise a cloud computing resource, a grid computing resource, and/or any other distributed computing arrangement. In some cases, the computing environment 103 can correspond to an elastic computing resource where the allotted capacity of processing, network, storage, or other computing-related resources can vary over time.

[0017] Various applications and/or other functionality can be executed in the computing environment 103 according to various embodiments. Also, various data is stored in a data store 112 that is accessible to the computing environment 103. The data store 112 can be representative of a plurality of data stores 112 as can be appreciated. The data stored in the data store 112, for example, is associated with the operation of the various applications and/or functional entities described below.

[0018] The components executed on the computing environment 103, for example, include a dispatch service 121 and other applications, services, processes, systems, engines, or functionality not discussed in detail herein. The dispatch service 121 is executed in order to facilitate the secure exchange of data among various client devices 106. To that end, the dispatch service 121 also performs

various backend functions associated with management and distribution of ciphertext data and associated cryptographic materials to the client devices 106 over the network 109. For example, the dispatch service 121 generates content pages such as, for example, web pages, multimedia messaging service (MMS) messages, and/or other types of network content that are provided to clients 106 for the purposes of facilitating secure storage and/or retrieval of data.

[0019] The data stored in the data store 112 includes, for example, a cryptographic application 131, application keys 132, user accounts 133, ciphertext records 139, and potentially other data. The cryptographic application 131 can be representative of a plurality of cryptographic applications 131 as can be appreciated. The cryptographic application 131 can be executable in the client device 106 to facilitate cryptographic services such as key generation, encryption, decryption, integrity checking, and/or other possible operations as can be appreciated. The cryptographic application 131 can implement various cryptographic algorithms necessary for these aforementioned services such as, for example, the data encryption standard (DES) algorithm, the advanced encryption standard (AES) algorithm, the triple data encryption standard (3-DES) algorithm, the Rivest, Shamir, Adelman (RSA) algorithm, the Twofish algorithm, the Threefish algorithm, the Blowfish algorithm, the Serpent algorithm, elliptic curve cryptography (ECC), various versions of the secure hash algorithm (SHA), the Skein hash algorithm, and/or other algorithms as can be appreciated.

[0020] The cryptographic application 131 can be directly executable by a processor of the client device 106 or by a virtual machine (*e.g.* Java®, JavaScript®, *etc.*) executing in the client device 106. In some embodiments, the cryptographic application 131 can include the ability to confirm the data integrity of the

cryptographic application 131 using techniques such as a digital signature, challenge-response handshake, client-side key verification, and/or other possible techniques. The application keys 132 can be representative of one or more cryptographic keys that can be associated with various instances of the cryptographic applications 131. In some embodiments, one or more of the application keys 132 can be usable to further validate a cryptographic application 131 executing in a client device 106 as will be described.

[0021] Each of the user accounts 133 can include information about a registered user of the dispatch service 121, such as, for example, name, address, email addresses, payment instruments, billing information, account settings, authentication credentials, user group membership, file management permissions, storage quotas and limitations, and/or other data. In some embodiments, the user accounts 133 can further include a public/private key pair comprising a public key 135 and a private key 136. The public/private key pair can be produced for use by implementations of RSA, ElGamal, ECC, Elliptic Curve Diffie-Helman ("ECDH") algorithm, or other asymmetric key ("public key") cryptography algorithms.

[0022] As the name suggests, the public key 135 can be publicly accessible to other users of the dispatch service 121, including users with and without a user account 133. The private key 136 can be protected by a cryptographic private wrapper 137. The private wrapper 137 can be generated according to AES key wrap specification, TDKW, PSEC-KEM, public key cryptography standards (PKCS), and/or other key wrap specifications as can be appreciated.

[0023] Each of the ciphertext records 139 include various data associated with ciphertext data provided by the client devices 106. The data included in each ciphertext record 139 can include ciphertext data 141, ciphertext metadata 143,

recipients 145, and/or other data associated with the cryptographic transformation of plaintext data obtained from the client 106.

[0024] The ciphertext data 141 includes the ciphertext produced from the plaintext by the cryptographic application 131 executing in the client device 106. In some embodiments, the ciphertext data 141 can include one or more pointers to other locations where the actual ciphertext is stored in segments or in the entirety. The ciphertext metadata 143 can include various metadata associated with the ciphertext data 141 such as, for example, one or more cryptographic hash values, the cryptographic algorithms used, access permissions, ownership identifiers, originator identifiers, and/or other possible metadata.

[0025] The recipients 145 of each ciphertext record 139 include the various parties for whom access to the ciphertext data 141 has been granted. Each of the recipients 145 can include a handle 146, a ciphertext key 147 secured by a recipient wrapper 148, and potentially other data. The handle 146 can include one or more identifiers through which the recipient 145 can access the ciphertext data 141. For example, the handle 146 can include a URI, uniform resource locator (URL), global unique identifier (GUID), and/or other types of identifiers as can be appreciated.

[0026] The ciphertext key 147 is the one or more keys used to configure the corresponding cryptographic application 131 used to generate the ciphertext data 141 associated with the given ciphertext record 139. The recipient wrapper 148 can be a cryptographic wrapper generated according to AES key wrap specification, TDKW, PSEC-KEM, public key cryptography standards (PKCS), and/or other key wrap specifications as can be appreciated. The ciphertext key 147 can be identical for all recipients 145 of a given ciphertext record 139, while the recipient wrapper 148 can be unique for each recipient 145. Each of the recipients 145 can further

include a log providing a history of access or attempts to access the ciphertext data 141, handle 146, ciphertext key 147, and/or other possible data.

[0027] Additionally, the data store can include one or more virtual file systems (VFS). The virtual file systems (VFS) can include various data associated with users or groups of users creating virtual file systems that use the ciphertext data 141 of one or more ciphertext records 139 for actual data storage. The virtual file system service can be implemented using the file system in userspace (FUSE) driver or other virtual file system driver as can be appreciated. The virtual file systems can further store the ciphertext metadata 143 associated with files stored in the virtual file systems which have been created. Audit records of the data store 112 can include a log of various activities undertaken with regard to the ciphertext records 139, contents of the virtual file systems, validation and execution of the cryptographic application 131 in the client device 106, and/or other interactions of the cryptographic application 131 with the dispatch service 121.

[0028] In some embodiments, the ciphertext metadata 143 stored in the virtual file systems can correspond to a master record and be encrypted by the ciphertext key 147. In such embodiments, a relational table can be associated with the master record in order to track file versions and/or file histories. File history records stored in the relational table can be separately encrypted with their own corresponding keys, permitting both encrypted metadata and searchable metadata to remain separate from individual file versions. Such embodiments allow multiple versions, such as multiple historical versions or other versions, of a file to be maintained in association with the master record.

[0029] The client 106 is representative of a plurality of client devices that can be coupled to the network 109. The client 106 can comprise, for example, a processor-

based system such as a computer system. Such a computer system can be embodied in the form of a desktop computer, a laptop computer, personal digital assistants, cellular telephones, smartphones, set-top boxes, music players, web pads, tablet computer systems, game consoles, electronic book readers, or other devices with like capability.

[0030] The client 106 can be configured to execute various applications such as a browser 161, virtual machine 163, and/or other applications, services, processes, systems, engines, or functionality not discussed in detail herein. The browser 161 can be executed in a client 106, for example, to initiate the cryptographic services offered by the computing environment 103 and/or other servers. The virtual machine 163 is a software implementation of a computer that is capable of executing the cryptographic application 131 and potentially other applications as would a physical computing device. Various virtual machines can be available on the client 106 including, for example, Java®, JavaScript®, Python, and/or other virtual machines as can be appreciated. In some embodiments, the virtual machine 163 can be integrated within the browser 161.

[0031] Next, a general description of the operation of the various components of the networked environment 100 is provided. To begin, an extended validation mechanism can be desirable for cryptographic applications 131 executing in various client devices 106. In various embodiments, the extended validation can be carried out through the use of one or more application keys 132 that can be embedded within a given cryptographic application 131, shown as application key 132a. Each of the application keys 132 stored in the data store 112 and/or embedded within the cryptographic applications 131 can usable to configure cryptographic algorithms which can be implemented in a cryptographic application 131. The application keys

132 can be generated using a pseudorandom number generator (PRNG) or other sources of key entropy.

[0032] In some embodiments, the application key 132a can not be stored as a constant value in the cryptographic application 131, but rather can be represented by a function or other programmatic code that can be used to produce the value of the application key 132a. As a non-limiting example, each instance of a cryptographic application 131 can contain a unique application identifier. Rather than storing the constant value of the application key 132a in the corresponding cryptographic application 131, a function can be stored that produces the value of the application key 132a when the corresponding unique application identifier is used as input.

[0033] Regardless of the operation used to embed a selected application key 132a within a given instance of the cryptographic application 131, the instance of the cryptographic application 131, along with the embedded application key 132a can be stored in the data store 112. Furthermore, an identifier of the instance of the cryptographic application 131, as well as the value of the application key 132a embedded within, can be stored as metadata within the application keys 132 or elsewhere within the data store 112. Thereafter, the instance of the cryptographic application 131 can be available for delivery to a client 106.

[0034] Subsequently, a client 106 can possess data to be encrypted and securely stored. To that end, the client 106 can establish a communication session with the computing environment 103 using the browser 161 or other client application. In some embodiments, the computing environment 103 can supply one or more session credentials with which the client device 106 can authenticate the computing environment 103. The session credentials supplied by the computing

environment 103 can include a digital certificate, a shared secret key, client-side key verification, and/or other possible credentials as can be appreciated.

[0035] In addition to authentication, the session credentials can be used to facilitate encryption of the communication session, thereby providing some degree of both authentication and confidentiality of the data as it is exchanged between the computing environment 103 and client 106. Establishing a communication session can occur as part of a secure socket layer / transport layer security (SSL/TLS) negotiation. Furthermore, in some embodiments, the client 106 can also provide one or more session credentials with which the computing environment 103 can authenticate the client device 106, therein providing mutual authentication. It should be noted that any credentials exchanged during establishment of the communication session can be independent of the credentials employed for later use in the generation of ciphertext data 141. Once the communication session is established, the dispatch service 121 can transmit an instance of the cryptographic application 131 for execution in the client 106.

[0036] In various embodiments, the cryptographic application 131 executing in the client 106 can include an embedded application key 132a with which an extended validation of the cryptographic application 131 can be performed. The extended validation can begin with the dispatch service 121 estimating the round-trip time for communication between the computing environment 103 and the client 106. The measurement of the round-trip time can occur during the previously described authentication phase or can be a separate communication between the computing environment 103 and the client 106. Additionally, the instance of cryptographic application 131 executing in the client can extract the embedded application key 132a. As previously described, in some embodiments, the application key 132a can

need to first be derived through use of a function or other programmatic code of the instance the cryptographic application 131.

[0037] In order to employ the application key 132a to perform the extended validation, the dispatch service 121 can transmit data for validation ("validation data") to the cryptographic application 131 executing in the client 106. The validation data transmitted can be recognized by the executing cryptographic application 131 as initiation of an extended validation. Thus, the cryptographic application can proceed to perform the pre-programmed validation routines upon the validation data, with which a response can be generated for the dispatch service 121.

[0038] As a non-limiting example, the data transmitted by the dispatch service 121 can include two different strings: a first string that is encrypted and the second string having no encryption applied. A valid cryptographic application 131 can be programmed to use the application key 132a to both decrypt the first string and to encrypt the second string, then transmit the two strings back to the dispatch service in a response. If the dispatch service 121 receives the response having the expected form, the dispatch service can examine the two strings in the response. If the cryptographic application 131 of the client 106 is valid and configured with the correct application key 132a, the data of the two strings will be as expected by the dispatch service 121. The dispatch service 121 formulates the expectation based on the known operations to be performed by a valid cryptographic application 131 and the known application key 132a embedded within the particular instance of the cryptographic application 131. In the case that the response is as expected from the cryptographic application 131, the dispatch service 121 can transmit a confirmation message to the cryptographic application 131 or, alternatively, no express approval can be sent.

[0039] Continuing the example, in the case that a response is not received from the client 106 within the expected round-trip time plus an acceptable delay tolerance, the dispatch service can re-initiate the extended validation operation using a different cryptographic application 131 having a different application key 132a. For example, if the response is received earlier than the estimated round-trip time plus an acceptable tolerance, it can indicate that the connection between the client 106 and the computing environment 103 is subject to tampering, such as by a third party performing a man-in-the-middle attack. As another example, if the response is received later than the estimated round-trip time plus an acceptable tolerance, it can indicate that a timeout, such as one due to packet loss, has occurred. Further, if a response is received by the dispatch service 121 and the form of the response and/or the two strings received in the response are not as expected, the dispatch service 121 can transmit a terminate instruction to the cryptographic application 131. In the event that no response is received or an improper response is received, the dispatch service 121 can report the incident to a user, as well as log the incident within an audit log of the data store 112.

[0040] Upon establishing a communication session with the computing environment 103, the client 106 can provide the dispatch service 121 with a service request to encrypt data available to the client 106. The data to be encrypted can be referred to as "plaintext" data throughout the present disclosure. However, as one skilled in the art can appreciate, use of the term "plaintext" does not require the data to be in a text format (*e.g.* American standard code for information interchange (ASCII), Unicode, *etc.*), nor does it suggest the data has no other encryption presently applied. A unit of data can simply be referred to as plaintext if it is in a non-encrypted state relative to a pending cryptographic operation.

[0041] In response to the service request, the dispatch service can deliver the cryptographic application 131 via the network 109. In response to obtaining the cryptographic application 131 through the browser 161 or other client application, execution of the cryptographic application 131 within a virtual machine 163 can be initiated in the client 106. In some embodiments, the cryptographic application 131 can include the ability to confirm the data integrity of the cryptographic application 131 as it executes in the client 106 using techniques such as a digital signature, challenge-response handshake, client-side key verification, and/or other possible techniques as can be appreciated.

[0042] A user of the client 106 can interact with the cryptographic application 131 executing in the client 106 to initiate the encryption and storage operation. The cryptographic application 131 can begin by creating a cryptographically strong ciphertext key 147 suitable for use by the encryption algorithm implemented in the cryptographic application 131. A strong ciphertext key 147 can be created using various sources of key entropy such as, for example, access time for a storage device, differences in the timing of the cores of a processor in the client 106, cryptographic-assistive hardware available to the client 106, and/or other possible sources. Thereafter, the requested encryption operation can be carried out by the cryptographic application 131 implementing one or more encryption algorithms configured with the ciphertext key 147. The product of the encryption operation is the ciphertext data 141.

[0043] In some embodiments, the cryptographic application 131 can calculate a cryptographic hash of the plaintext data prior to encryption. In various embodiments, the cryptographic hash value can be generated using a secret key associated with the computing environment 103, thereby creating a hash-based message

authentication code (HMAC). In other embodiments, the cryptographic hash value can be signed using with a private key of an asymmetric key pair, as is performed by implementations of the digital signature algorithm (DSA) or other possible algorithms providing digital signatures. Similarly, a cryptographic hash value of the ciphertext data 141 can also be produced. In some embodiments, the ciphertext data 141 can be divided into discrete segments from which the entire ciphertext data 141 can be reconstituted. In these embodiments, a cryptographic hash value can also be calculated for each individual segment of the ciphertext data 141.

[0044] The ciphertext data 141 produced by the cryptographic application 131 can be transmitted to the computing environment 103 for storage in a unique ciphertext record 139 of the data store 112. Additionally, the one or more cryptographic hashes computed on the plaintext data and the ciphertext data 141, including segments of the ciphertext data 141, can be placed in the ciphertext metadata 143 of the ciphertext record 139.

[0045] As described previously, the computing environment 103 can employ a plurality of computing devices. In light of this configuration, the ciphertext data 141, and/or segments thereof, can be stored in one or more computing devices of the computing environment 103. To affect the transfer of the ciphertext data 141 to the computing environment 103, the client device 106 can initiate one or more data transfer sessions to the computing environment 103 and/or the constituent computing devices of the computing environment 103. Throughout this disclosure, references of data transfer between the client 106 and the computing environment 103 should be understood to occur in light of various possible configurations. Furthermore, the data transfer can be undertaken using hypertext transfer protocol (HTTP), HTTP secure (HTTPS), file transfer protocol (FTP), trivial FTP (TFTP), file

service protocol (FSP), and/or other data transfer protocols, either connection-oriented or connectionless, as can be appreciated.

[0046] Independent of the transfer of the ciphertext data 141 to the data store 112, the cryptographic application 131 can encrypt the ciphertext key 147 with a recipient wrapper 148. The recipient wrapper 148 can be generated according to AES key wrap specification, TDKW, PSEC-KEM, public key cryptography standards (PKCS), and/or other key wrap specifications as can be appreciated. The recipient key used to generate the recipient wrapper 148 can be a shared secret, such as a passphrase, commonly known graphical shape, or a public key 135 associated with a user account 133 of the recipient.

[0047] This process can be repeated for each intended recipient for a given ciphertext data 141 resulting in identical copies of the ciphertext key 147 encrypted with recipient wrappers 148 that are unique to each recipient. Thereafter, each recipient wrapper 148, including the ciphertext key 147, can be transferred to the computing environment 103 for placement in a unique record for each recipient 145 of the ciphertext data 141. Furthermore, the dispatch service can generate a unique handle 146 through which each recipient 145 can access the ciphertext data 141 and their unique recipient wrapper 148. Once the encryption and storage operations requested by the user of the client 106 are complete, this portion of the cryptographic application 131 executing in the virtual machine 163 can terminate. In some embodiments, the cryptographic application 131 can overwrite and/or "zero-ize" any portions of residual data from operations of the cryptographic application 131 that can remain on the client device 106.

[0048] The dispatch service 121 can notify the various recipients 145 of the availability of data shared with them by an originator of the data. The notification can

be sent to an email address, instant message, short message service (SMS), MMS, and/or other methods of contact specified by the originator or included in user account 133 of a recipient. The notice sent to the contact address for each recipient 145 can include the handle 146 associated with the respective recipient 145. For example, the handle 146 can be a unique URI, wherein a user of a client 106 following the URI can initiate a communication session with the computing environment 103 using the browser 161 or other client application. As described previously, the client 106 can exchange various credentials with the computing environment in order to establish the communication session.

[0049] The handle 146 can serve as an embedded service request instructing the dispatch service that the client 106 requests access to particular ciphertext data 141 and recipient wrapper 148 associated with the recipient 145 for whom the handle 146 was created. In response to the service request, the dispatch service 121 can deliver the cryptographic application 131 via the network 109. In response to obtaining the cryptographic application 131 through the browser 161 or other client application, execution of the cryptographic application 131 within a virtual machine 163 can be initiated in the client 106.

[0050] In some embodiments, the ciphertext data 141 of one or more ciphertext records 139 can be shared among a group of users. In these embodiments, the cryptographic application 131 for the user creating the group can create a public/private key pair for the group, in addition to other keys that can be created for a ciphertext record 139 as described previously. In various embodiments, the group public/private key pair can be associated with a virtual file system or other type of virtual workspace that can be associated with one or more one or more ciphertext records 139. In these embodiments, the group public/private key pair for a virtual file

system can resemble a public/private key pair for a user account 133, and therefore can be considered a "virtual user."

[0051] The group public/private key pair can be produced by implementations of RSA, ElGamal, ECC, ECDH, or other asymmetric key ("public key") cryptography algorithms. The ciphertext key 147 can be encrypted using the group public key, resulting in a group wrapper for the ciphertext key 147. The group wrapper can be generated according to PSEC-KEM, public key cryptography standards (PKCS), and/or other asymmetric key wrap specifications as can be appreciated. The group wrapper, including the ciphertext key 147, can be stored in the ciphertext metadata 143 for the ciphertext record 139 and/or elsewhere within the data store 112 of the computing environment 103.

[0052] In these embodiments, the cryptographic application 131 can also encrypt the group private key with a recipient wrapper 148. The recipient wrapper 148 can be generated according to AES key wrap specification, TDKW, PSEC-KEM, public key cryptography standards (PKCS), and/or other key wrap specifications as can be appreciated. The recipient key used to generate the recipient wrapper 148 can be a shared secret, such as a passphrase, a commonly known graphical shape, or a public key 135 associated with a user account 133 of the recipient. This process can be repeated for each intended recipient (*i.e.* each group member) for a given ciphertext data 141, resulting in identical copies of the group private key encrypted with recipient wrappers 148 that are unique to each recipient. Thereafter, each recipient wrapper 148, including the group private key, can be transferred to the computing environment 103 for placement in a unique record for each recipient 145 of the ciphertext data 141.

[0053] A user of the client 106 can interact with the cryptographic application 131 executing in the client 106 to attempt a decryption and storage operation. The cryptographic application 131 can begin by obtaining both the ciphertext data 141 and recipient wrapper 148, embedded with the encrypted ciphertext key 147. In some embodiments, the ciphertext data 141 can be compared to one or more associated cryptographic hash values from the ciphertext metadata 143 in order to validate the data integrity of the ciphertext data 141 as received.

[0054] The recipient user can provide the cryptographic application 131 with their unique recipient key with which the ciphertext key 147 can be decrypted. The recipient user can provide the cryptographic application 131 with their unique recipient key with which the ciphertext key 147 can be decrypted. As discussed previously, the ciphertext key 147 can have been encrypted in the unique recipient wrapper 148 using a passphrase or the public key 135 associated with a user account 133 of the recipient user. In some embodiments, a key stretching and/or strengthening algorithm, such as the Password-Based Key Derivation Function 2 (PBKDF2), the bcrypt algorithm, the scrypt algorithm, and/or other such algorithms or approaches, can be used in conjunction with the passphrase to produce the unique recipient wrapper 148. In other embodiments, the ciphertext key 147 can have been encrypted in the unique recipient wrapper 148 using a graphical shape drawn by a user. In such embodiments, the graphical shape can be converted to a series of bits or bytes analogous to a passphrase, which can be used in conjunction with key stretching and/or strengthening algorithms such as PBKDF2, the bcrypt algorithm, the scrypt algorithm, and/or other such algorithms to encrypt the ciphertext key 147 in the unique recipient wrapper 148. As an illustrative and non-limiting example, the graphical shape can be used as an input to a hash algorithm and/or a key stretching

and/or strengthening algorithm to generate a key with an appropriate length and/or entropy to encrypt the ciphertext key 147 in the unique recipient wrapper 148. In some embodiments, additional information associated with the inputting of the graphical shape can be used as part of the input to the hash algorithm and/or the key stretching and/or strengthening algorithm. For example, the speed at which the graphical shape is drawn, pauses in drawing the graphical shape, and other information associated with inputting the graphical shape, can be used as part of the input to the hash algorithm and/or the key stretching and/or strengthening algorithm.

[0055] Further, such embodiments can combine the user of the passphrase and the graphical shape, as described above, to encrypt the ciphertext key 147 in the unique recipient wrapper 148. For example, the series of bits or bytes representing the passphrase and the graphical shape can be combined into a single input for a hash algorithm and/or a key stretching and/or strengthening algorithm to generate a key with an appropriate length and/or entropy to encrypt the ciphertext key 147 in the unique recipient wrapper 148. In such embodiments, the passphrase and the graphical shape can be concatenated together. In other embodiments, bits or bytes of the passphrase can be interspersed among the bits or bytes of the graphical shape or vice versa. In other embodiments, a first round of encryption of the ciphertext key 147 using the passphrase and a second round of encryption of the ciphertext key 147 using the graphical shape, or vice versa, can be performed to encrypt the ciphertext key 147 in the unique recipient wrapper 148.

[0056] Therefore, in order to decrypt the ciphertext key 147, the recipient user must enter the complementary credential used to perform the encryption. If a passphrase was used to generate the recipient wrapper 148 by the originating user, the same passphrase must be entered into the cryptographic application 131 by the

recipient user. If a graphical shape was used to generate the recipient wrapper 148 by the originating user, the same graphical shape must be drawn by the recipient user and submitted to the cryptographic application 131 by the recipient user.

[0057] Alternatively, if the originating user generated the recipient wrapper 148 using the public key 135 of the recipient user, the associated private key 136 must be used to decrypt the ciphertext key 147. In order for the recipient user to employ their own private key 136, it must first be decrypted from the private wrapper 137. Prior to performing the decryption of the private key 136, the private wrapper 137, including the encrypted private key 136, is obtained from the computing environment 103 by the cryptographic application 131. The recipient user supplies their personal passphrase or other user credential to decrypt their private key 136 from the private wrapper 137 within the context of the cryptographic application 131. Once the private key 136 is available, it can be applied to the cryptographic application 131 in order to decrypt the ciphertext key 147 from the recipient wrapper 148.

[0058] In some embodiments, the ciphertext record 139 accessed by the cryptographic application 131 can be shared by a group of users and can further be one of potentially many objects of a VFS or other virtual workspace. In these embodiments, the recipient wrapper 148 cannot include the ciphertext key 147, but instead a group private key. However, the same operations previously described with extracting a key from a recipient wrapper 148 can be applied whether the extracted key is a ciphertext key 147 or a group private key. Once the group private key is extracted from the recipient wrapper 148, the cryptographic application 131 can also obtain the group wrapper from the ciphertext metadata 143 or elsewhere within the data store 112. In these embodiments, the ciphertext key 147 can be

decrypted from within the group wrapper using the group private key extracted from the recipient wrapper 148.

[0059] Regardless of the method used to encrypt the ciphertext key 147, once it is obtained, the ciphertext key 147 can be applied to the cryptographic application 131 in order to decrypt the plaintext data from the ciphertext data 141 provided by the originating user. In some embodiments, the plaintext data can be compared to one or more associated cryptographic hash values, including any HMAC, from the ciphertext metadata 143 in order to validate the data integrity of the plaintext data. The validation can confirm that the plaintext data as decrypted by the cryptographic application 131 of the recipient user is the same as the plaintext data as encrypted by the cryptographic application 131 of the originating user.

[0060] In embodiments in which a ciphertext record 139 can be shared by a group of users, a member of the group can access and modify the ciphertext data 141 of the ciphertext record 139. In these embodiments, the ciphertext data 141 can be decrypted using the cryptographic application 131 as previously described. Thereafter, if modifications were made to the plaintext form of the ciphertext data 141, the modified version of the plaintext data can be encrypted and stored as ciphertext data 141 in the ciphertext record 139. In various embodiments, a new ciphertext key 147 can be generated and used to encrypt the modified plaintext data. The new ciphertext key 147 for the ciphertext data 141 can then be encrypted using the group public key, resulting in a new group wrapper for the new ciphertext key 147.

[0061] As discussed previously, the dispatch service 121 can further log various data associated with access or attempted access of the handle 146 and recipient wrapper 148 within a record associated with each recipient 145 and/or in the audit

records . Similarly, the cryptographic application 131 can notify the dispatch service 121 of the state of various events associated with attempts to decrypt the ciphertext data 141 such as, for example, mismatched cryptographic hash values, matching cryptographic hash values, incorrect recipient keys entered, and/or other possible events as can be appreciated. Such a history of interactions for a given recipient user associated with a recipient 145 can be used to offer and enforce services associated with recipient access such as a maximum number of failed decryption attempts, a maximum number of successful decryption attempts, and/or other services as can be appreciated.

[0062] Referring next to FIG. 2, shown is a flowchart that provides one example of the operation of a portion of the dispatch service 121 according to various embodiments. It is understood that the flowchart of FIG. 2 provides merely an example of the many different types of functional arrangements that can be employed to implement the operation of the portion of the dispatch service 121 as described herein. As an alternative, the flowchart of FIG. 2 can be viewed as depicting an example of steps of a method implemented in the computing environment 103 (FIG. 1) according to one or more embodiments.

[0063] This portion of the execution of the cryptographic application 131 can be executed in order to embed the application key 132a (FIG. 1) within an instance of the cryptographic application 131 (FIG. 1) to be stored for later execution in a client 106 (FIG. 1). Beginning with block 203, the dispatch service 121 can generate one or more application keys 132 using a pseudorandom number generator (PRNG) or other sources of key entropy. Next, in block 206, the dispatch service 121 can select one or more application keys 132 to be embedded within a given instance of a cryptographic application 131, shown as application key 132a. As described

previously, in some embodiments, the application key 132a can not be stored as a constant value, but rather can be represented by a function or other programmatic code that can be used to produce the application key 132a. Additionally, the dispatch service 121 can obfuscate all or portion of the code of the cryptographic application 131 using various possible code obfuscation operations as can be appreciated.

[0064] Then, in block 209, the dispatch service can employ other validation techniques for the cryptographic application 131 such as a digital signature, client-side key verification, and/or other possible techniques as can be appreciated. Subsequently, in block 212, the dispatch service 121 can store the instance of the cryptographic application 131, along with the embedded application key 132a, in the data store 112. Additionally, the dispatch service 121 can store the value of the application key 132a and an identifier for the particular instance of the cryptographic application 131 in the data store 112. Thereafter, this portion of the execution of the dispatch service ends as shown.

[0065] Turning now to FIG. 3, shown is a flowchart that provides another example of the operation of a portion of the dispatch service 121 according to various embodiments. It is understood that the flowchart of FIG. 3 provides merely an example of the many different types of functional arrangements that can be employed to implement the operation of the portion of the dispatch service 121 as described herein. As an alternative, the flowchart of FIG. 3 can be viewed as depicting an example of steps of a method implemented in the computing environment 103 (FIG. 1) according to one or more embodiments.

[0066] This portion of the execution of the cryptographic application 131 can be executed in response to a request from a client 106 (FIG. 1) to access and decrypt

data provided to them using a handle 146 (FIG. 1). Beginning with block 303, the dispatch service 121 can transmit an instance of the cryptographic application 131 for execution in the client 106. The cryptographic application 131 executing in the client 106 can include an embedded application key 132a with which an extended validation of the cryptographic application 131 can be performed. Then, in block 306, the dispatch service 121 can begin communicating with the cryptographic application 131 executing in the client 106.

[0067] Next, in block 309, the dispatch service 121 estimating the round-trip time for communication between the computing environment 103 and the client 106. Continuing, in block 312, the dispatch service 121 can generate data to be operated upon by the cryptographic application 131 executing in the client 106 that will be used as the basis for validation ("validation data"). Additionally, the dispatch service 121 can generate the expected response data for a valid cryptographic application 131 based at least upon the embedded application key 132a.

[0068] Then, in block 315, the dispatch service 121 can transmit the validation data to the cryptographic application 131 executing in the client 106. Next, in block 318, the dispatch service 121 determines if a timeout has occurred for a response from the cryptographic application 131. If a response is not received from the cryptographic application 131 within the estimated round-trip time plus an acceptable tolerance (*e.g.*. a timeout has occurred), then execution of the dispatch service 121 can return to block 303. For example, if the response is received earlier than the estimated round-trip time plus an acceptable tolerance, it can indicate that the connection between the client 106 and the computing environment 103 is subject to tampering, such as by a third party performing a man-in-the-middle attack. As another example, if the response is received later than the estimated round-trip time

plus an acceptable tolerance, it can indicate that a timeout, such as one due to packet loss, has occurred. Alternatively, if a time out has not yet occurred execution of the dispatch service proceeds to block 321.

[0069] At block 321, the dispatch service 121 determines if any response is received from the cryptographic application. If not response is yet received, execution of the dispatch service returns to block 318. Alternatively, if a response has been received, execution of the dispatch service 121 proceeds to block 324. At block 324, the dispatch service 121 determines if the received response is as expected for a valid cryptographic application 131.As a non-limiting example, the data transmitted by the dispatch service 121 can include two different strings, a first string that is encrypted and the second string having no encryption applied. A valid cryptographic application 131 can be programmed to use the application key 132a to both decrypt the first string and to encrypt the second string, then transmit the two strings back to the dispatch service in a response. If the dispatch service 121 receives the response having the expected form, the dispatch service can examine the two strings in the response. If the cryptographic application 131 of the client 106 is valid and configured with the correct application key 132a, the data of the two strings will be as expected by the dispatch service 121.

[0070] If, in block 327, a response is received by the dispatch service 121 and the form of the response is not as expected, the dispatch service 121 can transmit a terminate instruction to the cryptographic application 131. Alternatively, if the response is as expected for a valid cryptographic application 131, in block 330 results of the validation can be logged. Thereafter, this portion of the execution of the dispatch service 121 can end as shown.

[0071] With reference to FIG. 4, shown is a schematic block diagram of the computing environment 103 according to an embodiment of the present disclosure. Each computing environment 103 includes at least one processor circuit, for example, having a processor 403 and a memory 406, both of which are coupled to a local interface 409. The local interface 409 can comprise, for example, a data bus with an accompanying address/control bus or other bus structure as can be appreciated.

[0072] Stored in the memory 406 are both data and several components that are executable by the processor 403. In particular, stored in the memory 406 and executable by the processor 403 are the dispatch service 121, and potentially other applications. Also stored in the memory 406 can be a data store 112 and other data. In addition, an operating system can be stored in the memory 406 and executable by the processor 403.

[0073] It is understood that there can be other applications that are stored in the memory 406 and are executable by the processor 403 as can be appreciated. Where any component discussed herein is implemented in the form of software, any one of a number of programming languages can be employed such as, for example, C, C++, C#, Objective C, Java®, JavaScript®, Perl, PHP, Visual Basic®, Python®, Ruby, Flash®, or other programming languages.

[0074] A number of software components are stored in the memory 406 and are executable by the processor 403. In this respect, the term "executable" means a program file that is in a form that can ultimately be run by the processor 403. Examples of executable programs can be, for example, a compiled program that can be translated into machine code in a format that can be loaded into a random access portion of the memory 406 and run by the processor 403, source code that can be

expressed in proper format such as object code that is capable of being loaded into a random access portion of the memory 406 and executed by the processor 403, or source code that can be interpreted by another executable program, such as a virtual machine, to generate instructions in a random access portion of the memory 406 to be executed by the processor 403, *etc.* An executable program can be stored in any portion or component of the memory 406 including, for example, random access memory (RAM), read-only memory (ROM), hard drive, solid-state drive, USB flash drive, memory card, optical disc such as compact disc (CD) or digital versatile disc (DVD), floppy disk, magnetic tape, or other memory components.

[0075] The memory 406 is defined herein as including both volatile and nonvolatile memory and data storage components. Volatile components are those that do not retain data values upon loss of power. Nonvolatile components are those that retain data upon a loss of power. Thus, the memory 406 can comprise, for example, random access memory (RAM), read-only memory (ROM), hard disk drives, solid-state drives, USB flash drives, memory cards accessed via a memory card reader, floppy disks accessed via an associated floppy disk drive, optical discs accessed via an optical disc drive, magnetic tapes accessed via an appropriate tape drive, and/or other memory components, or a combination of any two or more of these memory components. In addition, the RAM can comprise, for example, static random access memory (SRAM), dynamic random access memory (DRAM), or magnetic random access memory (MRAM) and other such devices. The ROM can comprise, for example, a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electrically erasable programmable read-only memory (EEPROM), or other like memory device.

[0076] Also, the processor 403 can represent multiple processors 403 and/or multiple processor cores and the memory 406 can represent multiple memories 406 that operate in parallel processing circuits, respectively. In such a case, the local interface 409 can be an appropriate network that facilitates communication between any two of the multiple processors 403, between any processor 403 and any of the memories 406, or between any two of the memories 406, *etc.* The local interface 409 can comprise additional systems designed to coordinate this communication, including, for example, performing load balancing. The processor 403 can be of electrical or of some other available construction.

[0077] Although the dispatch service 121, and other various systems described herein can be embodied in software or code executed by general purpose hardware as discussed above, as an alternative the same can also be embodied in dedicated hardware or a combination of software/general purpose hardware and dedicated hardware. If embodied in dedicated hardware, each can be implemented as a circuit or state machine that employs any one of or a combination of a number of technologies. These technologies can include, but are not limited to, discrete logic circuits having logic gates for implementing various logic functions upon an application of one or more data signals, application specific integrated circuits (ASICs) having appropriate logic gates, field-programmable gate arrays (FPGAs), or other components, *etc.* Such technologies are generally well known by those skilled in the art and, consequently, are not described in detail herein.

[0078] The flowcharts of FIGS. 2 and 3 show the functionality and operation of an implementation of different portions of the dispatch service 121. If embodied in software, each block can represent a module, segment, or portion of code that comprises program instructions to implement the specified logical function(s). The

program instructions can be embodied in the form of source code that comprises human-readable statements written in a programming language or machine code that comprises numerical instructions recognizable by a suitable execution system such as a processor 403 in a computer system or other system. The machine code can be converted from the source code, *etc.* If embodied in hardware, each block can represent a circuit or a number of interconnected circuits to implement the specified logical function(s).

[0079] Although the flowcharts of FIGS. 2 and 3 show a specific order of execution, it is understood that the order of execution can differ from that which is depicted. For example, the order of execution of two or more blocks can be scrambled relative to the order shown. Also, two or more blocks shown in succession in FIGS. 2 and 3 can be executed concurrently or with partial concurrence. Further, in some embodiments, one or more of the blocks shown in FIGS. 2 and 3 can be skipped or omitted. In addition, any number of counters, state variables, warning semaphores, or messages might be added to the logical flow described herein, for purposes of enhanced utility, accounting, performance measurement, or providing troubleshooting aids, *etc.* It is understood that all such variations are within the scope of the present disclosure.

[0080] Also, any logic or application described herein, including the dispatch service 121, that comprises software or code can be embodied in any non-transitory computer-readable medium for use by or in connection with an instruction execution system such as, for example, a processor 403 in a computer system or other system. In this sense, the logic can comprise, for example, statements including instructions and declarations that can be fetched from the computer-readable medium and executed by the instruction execution system. In the context of the

present disclosure, a "computer-readable medium" can be any medium that can contain, store, or maintain the logic or application described herein for use by or in connection with the instruction execution system.

[0081] The computer-readable medium can comprise any one of many physical media such as, for example, magnetic, optical, or semiconductor media. More specific examples of a suitable computer-readable medium would include, but are not limited to, magnetic tapes, magnetic floppy diskettes, magnetic hard drives, memory cards, solid-state drives, USB flash drives, or optical discs. Also, the computer-readable medium can be a random access memory (RAM) including, for example, static random access memory (SRAM) and dynamic random access memory (DRAM), or magnetic random access memory (MRAM). In addition, the computer-readable medium can be a read-only memory (ROM), a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electrically erasable programmable read-only memory (EEPROM), or other type of memory device.

[0082] It should be emphasized that the above-described embodiments of the present disclosure are merely possible examples of implementations set forth for a clear understanding of the principles of the disclosure. Many variations and modifications can be made to the above-described embodiment(s) without departing substantially from the spirit and principles of the disclosure. All such modifications and variations are intended to be included herein within the scope of this disclosure and protected by the following claims.

CLAIMS

We claim:

1. A system, comprising:

a computing device; and

an application executable in the computing device, the application comprising:

logic that transmits a cryptographic application to a client device;

logic that establishes a communication link with the transmitted cryptographic application, wherein the transmitted cryptographic application is executing in the client device subsequent to transmission;

logic that generates validation data and expected response data, wherein the validation data and the expected response data are based at least in part on a cryptographic key embedded in the transmitted cryptographic application executing in in the client device; and

logic that sends the validation data to the transmitted cryptographic application executing in the client device.

2. The system of claim 1, wherein the application further comprises:

logic that measures a round-trip time for a communication with the transmitted cryptographic application executing in the client device, and

logic that detects whether a response is received from the transmitted cryptographic application executing in the client device within a time window comprising the measured round-trip time plus a delay tolerance.

3. The system of claim 2, wherein the application further comprises logic that retransmits the cryptographic application to the client device in response to detecting that the response is not received.

4. The system of claim 2 or 3, wherein the application further comprises logic that retransmits the cryptographic application to the client device in response to detecting that the response is received outside of the time window.

5. The system of any of claims 2-4, wherein the application further comprises:

logic that determines whether the response is valid based at least in part on a comparison of the response with the expected response data; and

logic that logs whether the response is valid.

6. The system of claim 5, wherein the application further comprises logic that sends a termination message to the transmitted cryptographic application executing in the client device in response to a determination that the response is invalid.

7. They system of claim 1, wherein the application further comprises:

logic that generates an application key;

logic that embeds the application key within the cryptographic application; and

logic that obfuscates at least a portion of the cryptographic application.

8. A method, comprising:

transmitting, via a computing device, a cryptographic application to a client device;

establishing, via the computing device, a communication link with the transmitted cryptographic application, wherein the transmitted cryptographic application is executing in the client device subsequent to transmission;

generating, via the computing device, validation data and expected response data, wherein the validation data and the expected response data are based at least in part on a cryptographic key embedded in the transmitted cryptographic application executing in in the client device; and

sending, via the computing device, the validation data to the transmitted cryptographic application executing in the client device.

9. The method of claim 8, further comprising:

measuring, via the computing device, a round-trip time for a communication with the transmitted cryptographic application executing in the client device, and

detecting, via the computing device, whether a response is received from the transmitted cryptographic application executing in the client device within a time window comprising the measured round-trip time plus a delay tolerance.

10. The method of claim 9, further comprising retransmitting, via the computing device, the cryptographic application to the client device in response to detecting that the response is not received.

11. The method of claim 9 or 10, further comprising retransmitting, via the computing device, the cryptographic application to the client device in response to detecting that the response is received outside of the time window.

12. The method of any of claims 9-11, further comprising:

determining, via the computing device, whether the response is valid based at least in part on a comparison of the response with the expected response data; and

logging, via the computing device, a result of determining whether the response is valid.

13. The method of claim 12, further comprising transmitting, via the computing device, a termination message to the transmitted cryptographic application executing in the client device in response to determining that the response is invalid.

14. The method of any of claims 8-13, further comprising:

generating, via the computing device, an application key;

embedding, via the computing device, the application key within the cryptographic application; and

obfuscating, via the computing device, at least a portion of the cryptographic application.

15. A non-transitory computer readable medium including a program comprising:

code that transmits a cryptographic application to a client device;

code that establishes a communication link with the transmitted cryptographic application, wherein the transmitted cryptographic application is executing in the client device subsequent to transmission;

code that generates validation data and expected response data, wherein the validation data and the expected response data are based at least in part on a cryptographic key embedded in the transmitted cryptographic application executing in in the client device; and

code that sends the validation data to the transmitted cryptographic application executing in the client device.

16. The non-transitory computer readable medium of claim 15, the program further comprising:

code that measures a round-trip time for a communication with the transmitted cryptographic application executing in the client device, and

code that that detects whether a response is received from the transmitted cryptographic application executing in the client device within a time window comprising the measured round-trip time plus a delay tolerance.

17. The non-transitory computer readable medium of claim 16, the program further comprising code that retransmits the cryptographic application to the client device in response to detecting that the response is not received.

18. The non-transitory computer readable medium of claim 16 or 17, the program further comprising code that retransmits the cryptographic application to the client device in response to detecting that the response is received outside of the time window.

19. The non-transitory computer readable medium of any of claims 16-18, the program further comprising:

code that determines that the response is valid based at least in part on a comparison of the response with the expected response data; and

code that logs whether the response is valid.

20. The non-transitory computer readable medium of claim 19, the program further comprising code that sends a termination message to the transmitted cryptographic application executing in the client device in response to a determination that the response is invalid.

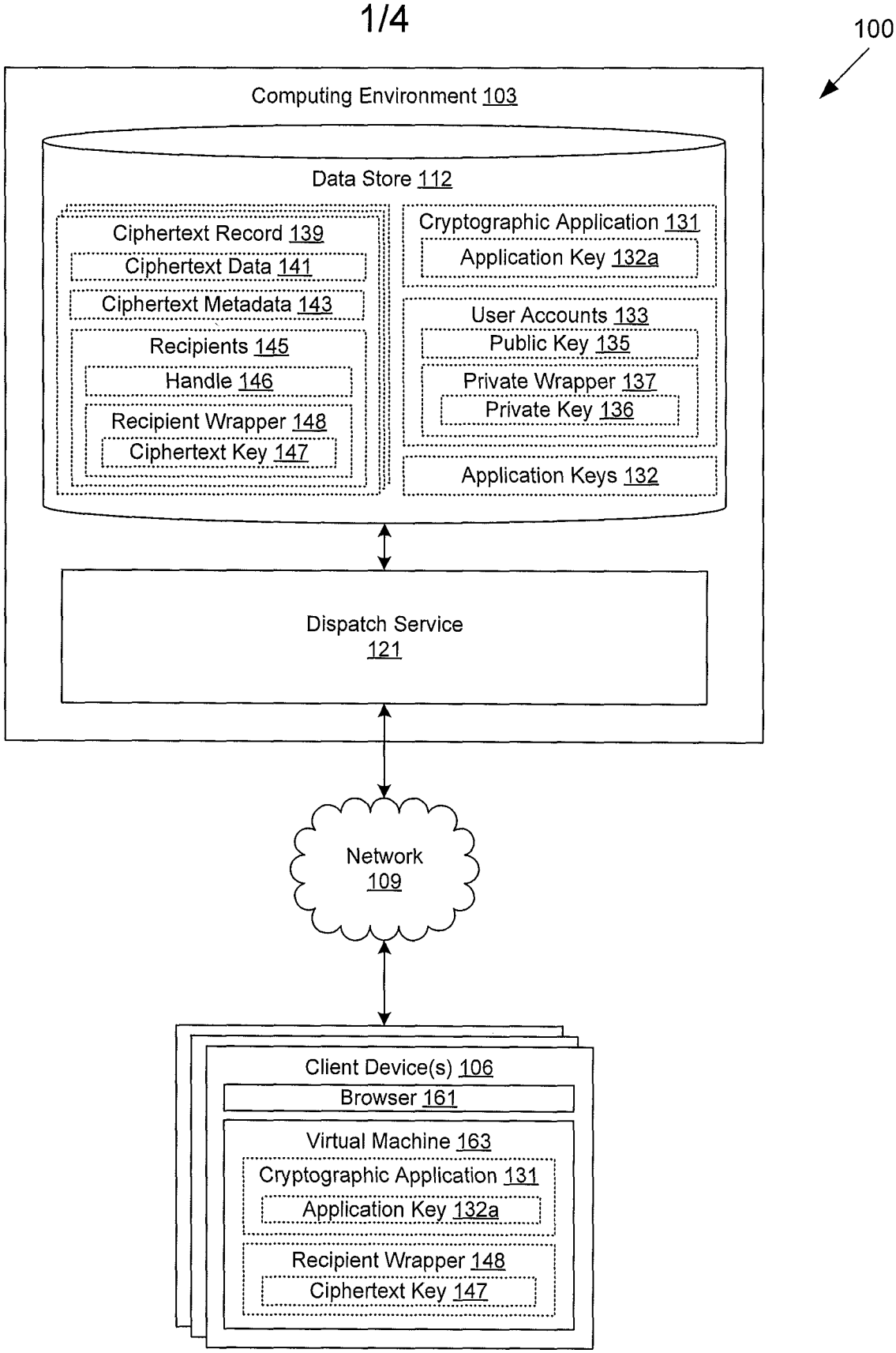
100
Computing Environment 103
Data Store 112
Ciphertext Record 139
Ciphertext Data 141
Ciphertext Metadata 143
Recipients 145
Handle 146
Recipient Wrapper 148
Ciphertext Key 147
Cryptographic Application 131
Application Key 132a
User Accounts 133
Public Key 135
Private Wrapper 137
Private Key 136
Application Keys 132
Dispatch Service 121
Network 109
Client Device(s) 106
Browser 161
Virtual Machine 163
Cryptographic Application 131
Application Key 132a
Recipient Wrapper 148
Ciphertext Key 147

FIG. 1

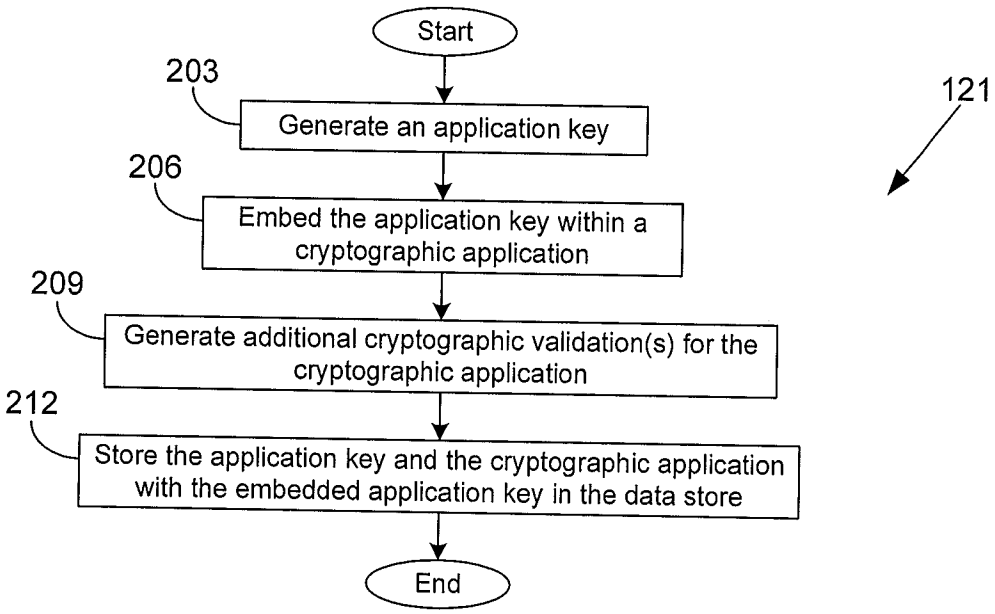
Start
203
Generate an application key
206
Embed the application key within a cryptographic application
209
Generate additional cryptographic validation(s) for the cryptographic application
212
Store the application key and the cryptographic application with the embedded application key in the data store
End
121

FIG. 2

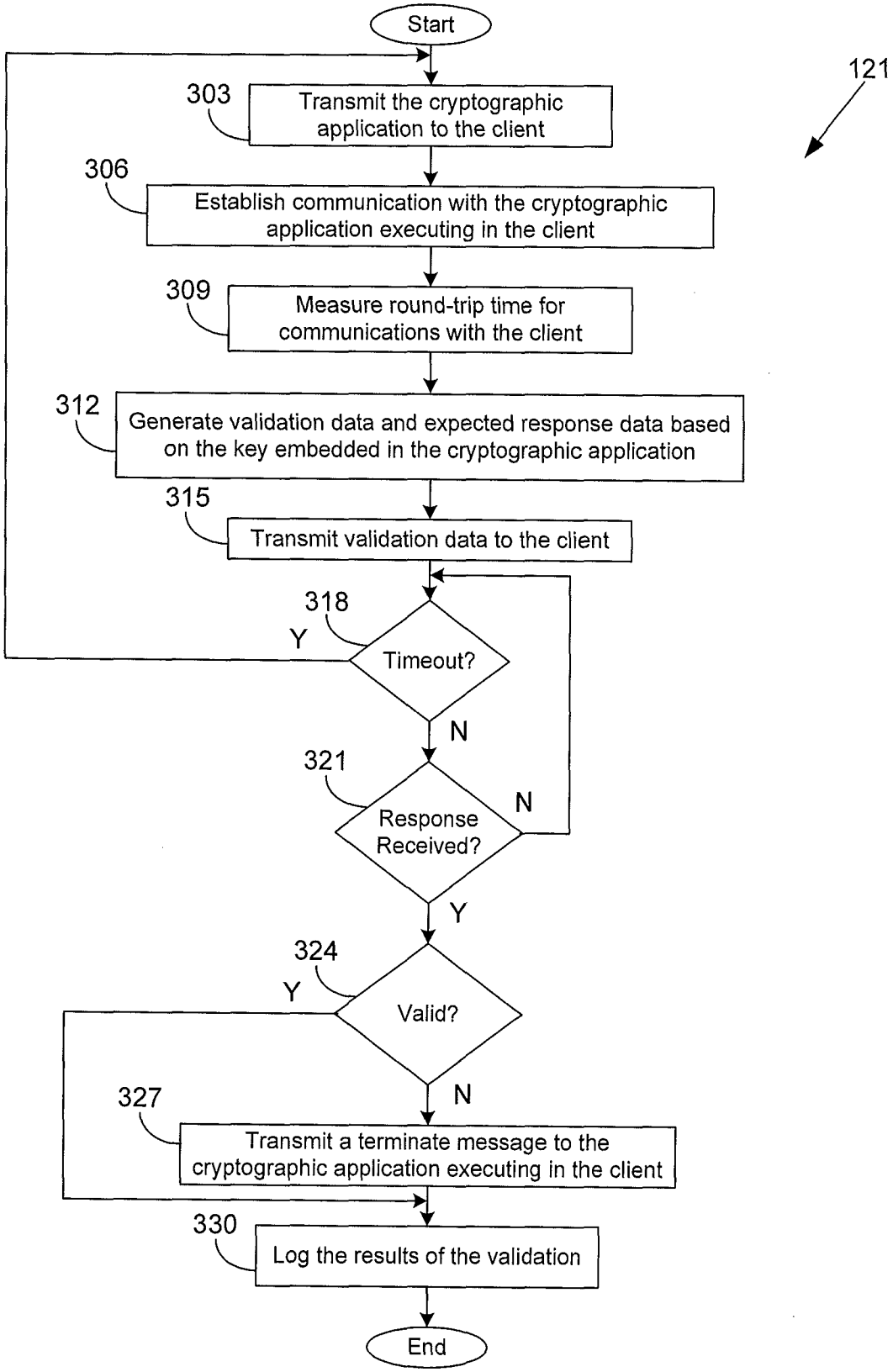
Start
121
303
Transmit the cryptographic application to the client
306
Establish communication with the cryptographic application executing in the client
309
Measure round-trip time for communications with the client
312
Generate validation data and expected response data based on the key embedded in the cryptographic application
315
Transmit validation data to the client
318
Y
Timeout?
N
321
Response Received?
N
Y
324
Y
Valid?
N
327
Transmit a terminate message to the cryptographic application executing in the client
330
Log the results of the validation
End

FIG. 3

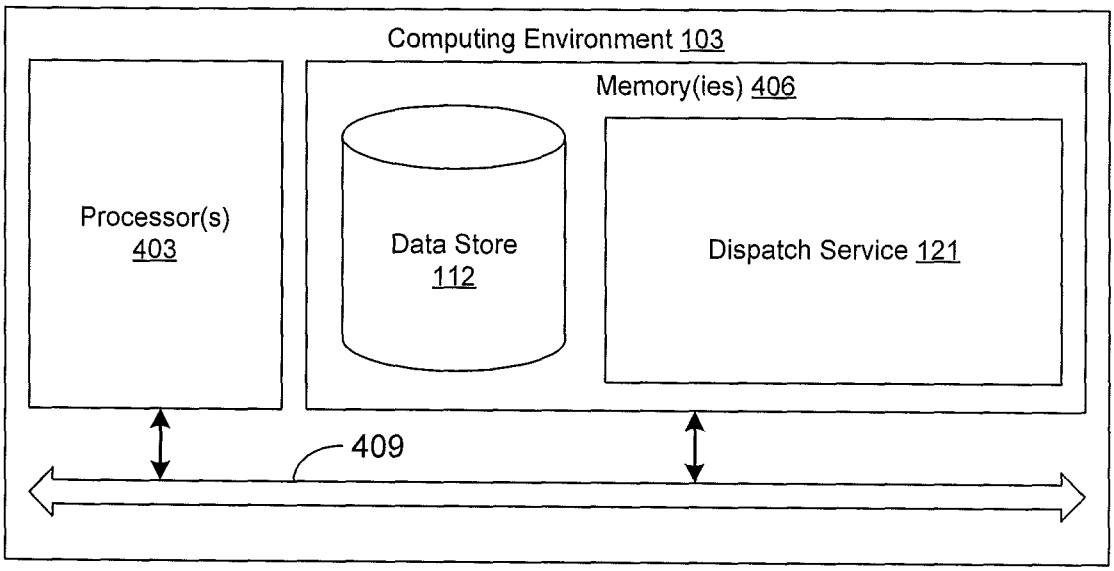
Computing Environment 103
Memory(ies) 406
Processor(s) 403
Data Store 112
Dispatch Service 121
409

FIG. 4

INTERNATIONAL SEARCH REPORT

PCT/US2013/078213 27.05.2014

International application No.
PCT/US13/78213

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - G06F 21/00; H04L 9/00, 9/30 (2014.01)

USPC - 713/161, 173, 189

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC(8): G06F 21/00, 21/60; H04L 9/00, 9/30 (2014.01)
USPC: 380/59; 713/161, 173, 189

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

MicroPatent (US-G, US-A, EP-A, EP-B, WO, JP-bib, DE-C,B, DE-A, DE-T, DE-U, GB-A, FR-A); Google; Google Scholar; ProQuest; IP.com; SEARCH TERMS: cryptographic application, program, software, validate, embeded, asymmetric key, secret key, private key, public key, ephemeral key, session key, expected response, timeout, packet loss, packet drop.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 2008/0120503 A1 (KIME, G., et al.) May 22, 2008; paragraphs [0027], [0029], [0031], [0036], [0038], [0042]-[0044].	1-4, 8-11, 15-18 --- 7
Y	US 2011/0307703 A1 (OGG, C., et al.) December 15, 2011; paragraphs [0046], [0049], [0057].	7
A	US2009/0313470 A1 (BADE, S., et al.) December 17, 2009; entire document.	1-4, 7-11, 15-18
A	US 2012/0131354 A1 (FRENCH, G.) May 24, 2012; entire document.	1-4, 7-11, 15-18

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 May 2014 (17.05.2014)	27 MAY 2014
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Shane Thomas PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

International application No.
PCT/US13/78213

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☒ Claims Nos.: 5-6, 12-14, and 19-20
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.