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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 October 2009 (29.10.2009)

PCT

(10) International Publication Number
WO 2009/132046 A2

(51) International Patent Classification:
G06F 21/24 (2006.01)

(21) International Application Number:
PCT/US2009/0413 14

(22) International Filing Date:
2 1 April 2009 (21 .04.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/046,751 2 1 April 2008 (21 .04.2008) US

(71) Applicant *(for all designated States except US):* **NCIPHER CORPORATION LTD.** [GB/GB]; Jupiter House, Station Road, Cambridge CBl 2JD (GB).

(71) Applicant and
(72) Inventor: NOLL, Landon, Curt [US/US]; 964. 1 BeImont Terrace, Sunnyvale, CA 94086 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only):* **WINTER, Christo¬pher Norman** [US/US]; 19586 High Barbaree, Grass Valley, CA 95945-8700 (US).

(74) **Agents:** BONE, Richard, G. A. et al; Fish & Richardson P.C., P.O. Box 1022, Minneapolis, MN 55440-1022 (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, BR, BW, BY, BZ, CA, CH, 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, IS, JP, KE, KG, KM, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, **TJ,**

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR SECURITY REQUIRING AUTHORIZATION BY MULTIPLE USERS

(57) Abstract: Methods, systems, and apparatus, including computer program products, for security requiring authorization by multiple users. In one embodiment, one or more data files are encrypted by specifying an encryption policy, including an authorization group and a number of authorizations within the authorization group being required for accessing the output file. The authorization group is defined. The data files are encrypted in accordance with the policy. In another embodiment, a file is decrypted by receiving a file encrypted in accordance with an encryption policy including a first number of authorizations within an authorization group required for accessing the file, receiving authorization codes from members of the authorization group, determining a second number of correct authorization codes received, and decrypting the file if the second number is equal to or greater than the first number.

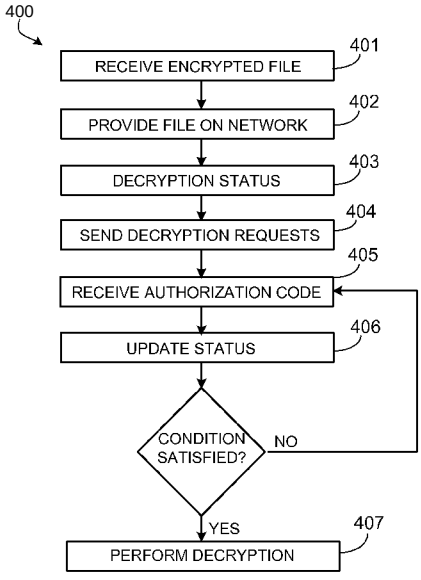

FIG. 4

WO 2009/132046 A2

TM), European (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, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

METHOD AND SYSTEM FOR SECURITY REQUIRINGAUTHORIZATION BY MULTIPLE USERS

CLAIM OF PRIORITY

[0001] The present application claims benefit of priority to U.S. provisional application serial no. 61/046,751, filed April 21, 2008, the disclosure of which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] This application relates generally to system and file security, and more specifically to controlling access to files for which there are multiple authorized users.

BACKGROUND

[0003] Information and its protection have been an important aspect of life. Society as a whole evolved through creating, using, and sharing information. Over time, information has become a more and more valuable resource. And, as with other resources, it is desirable to control access to information.

[0004] In the past, our ancestors devised various ways of protecting valuable and/or sensitive information. One of the most common forms of protection has been physical separation. For example, important documents of royal families are often stored (and even hidden) at a secure place, to which access is impossible to all but select members of royal families. While effective, this type of protection is burdensome for sharing of information, even for those who are entitled to the information. Another type of physical protection has been to use special seals and/or packages to make protected documents inaccessible in certain ways. However, these methods are unsatisfactory solutions at least because they are cumbersome.

[0005] In the last century, with the advent of telecommunication techniques, and in the events of two world wars, the protection of information has in large part been focused on protection of electronic information. For example, various coding schemes have been developed to protect radio

transmissions of military information. Specific code machines have been used by intelligence agencies.

[0006] In today's world, the protection of information is more important than ever. Much information, private or public, trivial or important, is often stored and communicated in the electronic form, which is far easier to duplicate than the physical forms of old. To protect valuable information, various types of information protection and/or information security techniques have been developed.

[0007] In general, information protection in this information age refers to protecting data from unauthorized access, use, disclosure, destruction, modification, or disruption. Among other things, the goal is to protect the confidentiality, integrity, and availability of information.

[0008] For digital information, one of the most common ways of protecting data is access control. That is, access to protected information is restricted to authorized persons.

[0009] Generally, conventional information protection techniques prevent unauthorized users from accessing protected information. For example, password protection and data encryption are common techniques. However, conventional techniques are often inadequate.

[0010] Therefore, an improved method and system for information protection is desired.

SUMMARY

[0011] This application relates generally to system and file security. More specifically, embodiments according to the present technology provide a technique for a security scheme, in which access to files or systems is granted only after a predetermined number of security measures has been taken. For example, wherein the security measure is an unlocking operation, the specific sequence of unlockings (e.g., providing correct access codes) is immaterial, but the number of unlockings performed determines whether the access is granted. In a specific embodiment, there are N keys that are associated with the access of a specific file, and the file only becomes accessible after a lesser number, M, of keys have been received. The technology provides particular benefit when both N and M are greater than 1. There are other embodiments as well.

[0012] According to an embodiment, the present technology provides a method for encrypting one or more data files. The method includes specifying an encryption policy for encrypting the one or

more data files and generating an output file. The encryption policy includes at least an authorization group and a number. The encryption policy specifies the number of authorizations within the authorization group being required for accessing the output file. The method further includes defining the authorization group, the authorization group including a plurality of users being able to satisfy the number. Each of the plurality of users has an authorization code. The method additionally includes encrypting the one or more data files in accordance with the encryption policy. Other embodiments include corresponding systems, apparatus, computer program products, and computer readable media.

[0013] According to another embodiment, the present technology provides a method for decrypting a file. The method includes receiving a file. The file is encrypted in accordance with an encryption policy. The encryption policy includes at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file. The method further includes receiving authorization codes from members of the authorization group. Additionally, the method includes determining a second number, the second number being a number of correct authorization codes received from members of the authorization group. Also, the method includes decrypting the file if the second number is equal to or greater than the first number. Other embodiments include corresponding systems, apparatus, computer program products, and computer readable media.

[0014] According to yet another embodiment, the present technology provides a method for decrypting a file. The method includes receiving a file. The file is encrypted in accordance with an encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file. The authorization group includes a plurality of users. The method also includes storing the file on storage, the storage being accessible by the plurality of users. The method also includes providing a status indicator associated with a decryption status for the file. In addition, the method includes sending a plurality of decryption requests to the plurality of users. Furthermore, the method includes receiving authorization codes from a second number of the plurality of users. The method also includes updating the status indicator based on the authorization codes received from the second number of the plurality of users. Additionally, the method includes decrypting the file if the second number is

equal to or greater than the first number. Other embodiments include corresponding systems, apparatus, computer program products, and computer readable media.

[0015] According to yet another embodiment, the present technology provides a method for encrypting plaintext data. The method includes encrypting the data with a first encryption key to generate at least an encrypted file and obtaining a number N of encryption codes, each of the encryption codes being associated with a user. The method further includes defining an encryption policy, the encryption policy specifying a number M, the number M being a specified number of encryption codes required for performing decryption. The method further includes generating at least N !/(M!*(N-M)!) number of combination keys, each of combination keys being a function of M number of encryption codes and the first encryption key, and storing the combination keys. The technology provides particular advantages when both N and M are greater than 1.

[0016] According to yet another embodiment, the present technology provides a method for accessing an encrypted file. The method includes receiving an encrypted file. The encrypted file includes a predetermined number of combination keys, each of the combination keys being a function of a number M of authorization codes. The method further includes requesting the authorization codes from a plurality of users chosen from a number N of users having one of the authorization codes, N being greater than M and, typically, both N and M being greater than 1. The method further includes receiving the authorization codes and verifying the authorization codes. The method further includes forming an unlocking key if the a number of verified authorization codes is equal to or exceeds the number M and decrypting the encrypted file using the unlocking key. Other embodiments include corresponding systems, apparatus, computer program products, and computer readable media.

[0017] According to yet another embodiment, the present technology provides a system for encrypting one or more files. The system includes a storage that is configured to store one or more data files. The system also includes a user interface that includes a display and an input device. The system further includes a processor. The user interface is configured to receive instructions' for specifying an encryption policy for encrypting the one or more data files and generating an output file. The encryption policy includes at least an authorization group and a number. The encryption policy specifies the number of authorizations within the authorization group being required for accessing the output file. The processor is configured for defining the authorization group, which

includes a plurality of users being able to satisfy the number. For example, each of the plurality of users has an authorization code. The processor is further configured for encrypting the one or more data files in accordance with the encryption policy. The storage is being configured to store the output file. Other embodiments include corresponding methods, apparatus, computer program products, and computer readable media.

[0018] It is to be appreciated various embodiments of the present technology provide numerous advantages over conventional techniques. In a specific embodiment, the present technology provides a flexible method that allows multiple users to determine the access of a protected file, which is useful in many ways, especially in large corporate settings. In addition, certain embodiments of the present technology are compatible with a wide range of existing systems. For example, a system according to the present technology is may be implemented as a software added on solution. There are other benefits as well.

[0019] Depending upon embodiment, one or more of these benefits may be achieved. These benefits and various additional objects, features and advantages of the present technology can be fully appreciated with reference to the detailed description and accompanying drawings that follow.

DESCRIPTION OF DRAWINGS

[0020] FIG.l is a diagram illustrating a data security system according to an embodiment described herein;

[0021] FIG. 2 is a diagram illustrating an encryption header according to an embodiment described herein;

[0022] FIG. 3 is a flow diagram illustrating a process for encrypting information according to an embodiment described herein;

[0023] FIG.4 is a diagram illustrating a decryption process according to an embodiment described herein;

[0024] FIG. 5 is a diagram illustrating an alternative process for access an encrypted file according to an embodiment described herein;

[0025] FIG. 6 is a diagram illustrating an encryption process according to an embodiment described herein; and

[0026] FIG. 7 is a diagram illustrating a decryption process according to an embodiment described herein.

DETAILED DESCRIPTION

[0027] This application relates generally to system and file security. More specifically, embodiments of the present technology provide a technique for a security scheme, in which access to files or systems is granted only after a predetermined number of security measures (e.g., unlocking operations) has been executed. For example, the specific sequence of unlockings (e.g., receiving correct access codes) is immaterial, but the number of unlockings performed determines whether the access is granted. In a specific embodiment, there are N keys associated with the access of a specific file, and the file only becomes accessible after a number, M (where M is less than N and, typically, both N and M exceed 1), of keys has been unlocked. There are other embodiments as well.

Introduction and Representative Hardware

[0028] It is desirable to provide an information protection technique in which access to data is allowed after having been authorized by a predetermined number of users. For example, in a corporate environment where information is constructed and shared by a group of people, it is often desired to have a scheme in which the right to access (and/or modify) is subjected to the approval of multiple users. There could be many similar scenarios.

[0029] It is to be appreciated that embodiments of the present technology have a wide range of applications in a variety of settings. In a specific embodiment, the present technology provides an information protection scheme in a storage area network, where various types of security measures have been taken to protect data from external security breaches. In certain embodiments, the present technology provides a technique where the data protection scheme is specifically tailored for one or more files, the access to which may be authorized by users both inside and outside a storage area network. It is to be understood that there are other embodiments as well.

[0030] FIG. 1 is a simplified diagram illustrating a data security system 100 according to an embodiment of the present technology. This diagram is merely an example, which should not

unduly limit the scope of the claims. One of ordinary skill in the art would recognize many variations, alternatives, and modifications.

[0031] As shown, system 100 includes a processing component 105, terminals 108 and 109, a storage area network 106, and a storage 110. It is to be understood that there may be other components in system 100 as well. For example, there can be additional terminals in addition to terminals 108 and 109.

[0032] The processing component 105 is configured to receive user inputs and to perform encryption. Depending on the application, the processing component 105 may be implemented in various ways. In a specific embodiment, the processing component 105 is a workstation in a computer network. In another embodiment, the processing component 105 is a portable computer. The processing component 105 includes, among other things, a display 101, a processor 102, a keyboard 103, and a mouse 104. The processing component 105, as shown is connected to the storage area network 106, for example, through a secured connection (e.g., secured high-speed wired connection). In certain embodiments, the processing component 105 is apart of the storage area network 106. Depending on the application, the processing component 105 may also include various types of storage media, such as hard disk, compact disc, backup tape drive, etc. For example, these storage media can be used to temporarily or permanently store various types of data, including the information that are to be encrypted and/or decrypted.

[0033] The processor component 105 is configured to receive user input and provide a display. For example, using the keyboard 103 and the mouse 104, a user is able to provide specific instructions associated with encrypting and/or decrypting data. In a specific embodiment, the processor component 105 includes a user interface that is configured to receive instructions for specifying an encryption policy for encrypting the one or more data files and generating an output file.

[0034] For example, the encryption policy includes, among other things, at least an authorization group and a number. The encryption policy specifies the number of authorizations within the authorization group that is required for accessing the output file. The processor component is also configured for defining the authorization group. For example, the authorization group includes a plurality of users being able to satisfy the number of authorization users. Depending on the application, the authorization group may be predetermined and/or defined on a case-by-case basis.

As merely an example, the authorization group is predefined to include members of a specific project who use and exercise control over the files that are encrypted. Each member may supply their own access codes, or the access codes are randomly generated and then provided to each of the members.

[0035] Depending on the application, various techniques according to the present technology are implemented by way of a software solution, e.g., a computer program product stored on a computer-readable medium and executed by one or more processors. A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, declarative or procedural languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, object, or other unit suitable for use in a computing environment. A computer program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub-programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network. For example, a user-friendly software interface can be provided so that users are able to provide various information related to the encryption and decryption of the files.

[0036] The processor component 105 is also used for the actual encryption. Depending on the application, various types of encryption methods may be used. For example, an encryption key is generated for the purpose of encrypting files. In addition, the encrypted file includes a file header that is based on the encryption policy. In addition to encryption, the processor component 105 may additionally compress files.

Representative Data File Structure

[0037] FIG. 2 is a simplified diagram illustrating an encryption header according to an embodiment of the present technology. This diagram is merely an example, which should not unduly limit the scope of the claims. One of ordinary skill in the art would recognize many variations, alternatives, and modifications.

[0038] As shown in FIG. 2, the header 200 includes the following fields:

1. file name;
2. decryption key;
3. access policy;
4. key status;
5. encryption verification;
6. keys; and
7. encryption method.

[0039] Various fields as shown in the header 200 may be added, removed, modified, and/or replaced, and should not limit the scope of technology. Typically, these fields are defined according to the encryption policy, predetermined or manually provided, and may vary from file to file.

[0040] The file name field, depending on the application, may include various types of information, such as location (link) of file(s) stored, and names of the files that are encrypted, and the names of encrypted files themselves. The file name may be displayed or hidden based on the encryption policy. For example, the file name field may be encrypted as well.

[0041] The decryption key field includes a decryption key for decrypting the encrypted file. In a specific embodiment, the decryption key is defined at the time when the encryption process is performed. For example, the decryption key includes a randomly generated nonce (e.g., encryption initialization character) that is used for the encryption, and the nonce is used for decryption of the file. According to embodiments, the decryption key allows access and decryption of the encrypted file. For example, the decryption key becomes available only after the predetermined number of users has approved the access.

[0042] The access policy field stores various types of access policies, and is based on the encryption policy. The access policy field includes information regarding the members of the authorization group and the number of authorizations required for accessing the encrypted file. According to embodiments, the number of authorizations required for access the file is less than the total number of members of the authorization group. It is to be appreciated that such a scheme provides various benefits, as it is possible to define a minimum quorum within a group that is required to authorize access. In addition, this scheme may be flexibly implemented based on embodiments of the present technology.

[0043] In addition, the access policy field may include different types of policies for different types of access. For example, the access policy may define one authorization group and a number required for viewing the encrypted content, and a different authorization group and a different number required for writing and/or modifying the encrypted content.

[0044] Furthermore, the access policy field may include information that further defines access groups. In a specific embodiment, the access policies for access may be different for different users. For example, for a CEO of a company to access the encrypted file, the requirement for authorization may be less stringent than for other members of the company.

[0045] In certain embodiments, the access policy field defines not only what authorization codes are needed, but also how the authorization codes are to be provided. In one embodiment, authorization codes may be provided in any order, and the predetermined number of authorization codes is needed. In another embodiment, the authorization codes are to be provided in specific orders, whether sequential or not, for the access to the encrypted file to be granted.

[0046] The key status field includes various information associated with the status of keys and/or authorization codes. It is to be understood that the terms "keys" and "authorization codes" are used interchangeably in the body of the specification and should not unduly limit the scope of technology or appended claims. The key status field contains, among other things, information related to the number of keys and/or authorization codes that have been received. In a specific embodiment, the key status field includes the number of keys and/or authorization codes received. In another embodiment, the key status field includes specific status associated with each authorization code, which includes, but not limited to, whether the particular key and/or authorization that has been received is correct, and/or the number of attempts of failure for providing the key and/or the authorization code. In addition, the key status field may also include information related to the order in which the keys and/or authorization codes have been received.

[0047] The verification field includes information related to verification for the encryption and/or decryption of the file. For example, the verification code is to ensure that the decryption process is correctly performed.

[0048] The keys field includes various keys and authorization codes. As explained above, the keys and/or authorization codes may be user provided, randomly generated, and/or the combination

thereof. For added security, the keys themselves may be encrypted within the header. Depending on the encryption policies, the number of keys stored varies. For example, the number of keys may be equal to the numbers of members within the authorization group (i.e., each member provides or is assigned a key). In a specific embodiment, each of the keys is associated with a particular member of the authorization group, and the information concerning that particular member is also stored in the keys field. In other embodiments, keys and members share a one-to-one correspondence that is randomly assigned.

[0049] In addition to the fields described elsewhere herein, the header 200 may include other fields and information as well. For example, the header 200 includes fields such as encryption method, encryption date, etc.

[0050] Now referring back to FIG. 1, the processor component 105, as explained herein, is configured to process and/or encrypt files. In addition, the processor component is connected to the storage area network 106. The terminals 108 and 109 are also connected to the storage area network 106. For example, the terminals 108 and 109 can be used by different users, who use these two terminals to provide customized authorization codes to the processor component during the encryption process. There may be additional terminals as well.

[0051] The processor component 105 is also connected to the storage 110 via the storage area network 106. It is to be understood that the processor component 105 can be an integral part of the storage area network 106. The storage area network 106, as an example, is a part of the internal network, which is protected by firewalls and various other types of security measures.

[0052] Merely by way of an example, the storage 110 is one of much storage on the storage area network 106. Depending on the application, the storage 110 may include various types of hardware storage components (i.e., computer readable media), such as hard disks, optical discs (e.g., CD, CD-RW, DVD-R), tape, etc. In addition, the storage 110 may be configured to securely store information using RAID drives. In a specific embodiment, the storage 110 stores the encrypted file, and the storage 110 is accessible to the processor component 105, and terminals 108 and 109. Merely as an example, the processor component 105 and the terminals (and possibly additional terminals as well) have access to the storage 110 during the decryption process.

Representative Methods

[0053] FIGS. 3, 4, and 5 show flow diagrams of processes according to embodiments of the present technology. These diagrams are merely examples, which should not unduly limit the scope of the technology or appended claims. One of ordinary skill in the art would recognize many variations, alternatives, and modifications. For example, various steps may be added, removed, replaced, rearranged, repeated, modified, overlapped, and/or partially overlapped, and should not unduly limit the scope of claims. As merely an example, the processes can be implemented in conjunction with the system 100.

[0054] FIG. 3 includes the following steps:

1. receive one or more data files (step 301);
2. specify an encryption policy (step 302);
3. obtain parameters for the encryption policy (step 303);
4. define authorization group and authorization codes (step 304);
5. process the one or more data files (step 305); and
6. store the encrypted (and/or compressed) file (step 306).

[0055] At step 301, data files that are to be encrypted are received. As an example, the data file is stored on a network storage device in a storage area network. Depending on the application, the one or more files may be grouped together for encryption. The one or more data files can be plaintext files and/or encrypted files.

[0056] At step 302, an encryption policy is specified. Depending on the application, the encryption policy may be predetermined and/or determined on a case-by-case basis. In a specific embodiment, the encryption policy is provided by the owner(s) of the one or more data files. In another embodiment, the encryption policy is provided by a system administrator of the storage area network. For example, the encryption policy includes information regarding the member of the authorization group and the number of authorizations required for accessing the encrypted file. According to some embodiments, the number of authorizations required for access to the file is less than the total number of members of the authorization group.

[0057] At step 303, parameters for the encryption policies are obtained. In an embodiment, the parameters are received. As explained elsewhere herein, the parameters may be received from one or more users manually (e.g., users entering passwords and/or keys). For example, the owner of the

one or more data files that are to be encrypted can provide a list of members in the authorization group and the number of authorization codes being required for accessing the encrypted file. In a specific embodiment, an interface such as a "user-friendly" interface is displayed to allow the user to enter this information. According to an alternative embodiment, parameters are based on predetermined encryption policies.

[0058] At step 304, the authorization group and authorization codes are defined. According to a specific embodiment, the authorization group is specified by the owner of the files, and the authorization codes are randomly generated and then provided to the members of the authorization group. According to another embodiment, the authorization group is defined according to a predetermined encryption policy, and each member of the authorization group is prompted by a message to enter her or his own authorization code.

[0059] At step 305, the one or more data files are processed. More specifically, the one or more files are encrypted. Depending on the application, the one or more data may also be compressed and/or subjected to other processes as well. During the encryption process, a header is generated and stored in accordance with a data structure format. For example, the header is illustrated in FIG. 2 and explained above. The encryption in step 305 may be various types of data encryption. For example, encryption key(s) of various lengths may be generated and used for the encryption process. Depending on the application, the encryption key itself can be encrypted in various methods, which are described below. In an embodiment, encryption keys are encrypted using authorization codes. For example, multiple encrypted encryption keys are generated as a function of the key used for encrypting the files and the authorization codes. In addition, the encryption key(s) may be stored in a hidden section of the encrypted file.

[0060] At step 306, the encrypted file is stored. According to an embodiment, the encrypted file is stored on a secured network storage device. For example, the encrypted file is stored on a dedicated network RAID device. According to another embodiment, the encrypted file is stored on a local computer.

[0061] FIG. 4 is a diagram illustrating a decryption process according to an embodiment of the present technology, and includes the following steps:

1. receive an encrypted file (401);
2. store the encrypted file on a network storage (402);

3. provide a decryption status indicator for the encrypted file (403);

4. send decryption request to users (404);

5. receive authorization codes from users (405);

6. update the status indicator (406); and

7. decrypt the encrypted file if condition satisfied (407).

[0062] At step 401, an encrypted file is provided. According to an embodiment, the encrypted file includes information (e.g., encryption keys, pointers) to other protected files. In another embodiment, the encrypted file contains an encryption key (and/or other information) that is needed to unlock one or more files. Depending on the application, the encrypted file may be compatible with various types of formats. For example, the encrypted file can include a header file which is substantially similar to the header shown in FIG. 2.

[0063] At step 402, the encrypted file is stored on network storage. The storage of the encrypted file on a network storage allows the encrypted file to be better protected against potential data loss due to various types of hardware failure. For example, the network storage where the encrypted file is stored is a RAID drive where duplicate copies of the encrypted file is stored to allow better reliability. In addition, being stored on the network drive, the encrypted file is accessible to different users who provide authorization codes for access of the encrypted file.

[0064] At step 403, a status indicator associated with the encrypted file is provided. For example, the status indicator is a part of the encrypted file itself. The status indicator is used to record the status related to the decryption of the encrypted file. Among other things, the status indicator records the number of times proper authorization codes have been received and whether the encrypted file is to be decrypted and/or accessed. For example, if the number of correct authorization codes is equal to or exceeds the predetermined threshold, the status indicator would indicate that the encrypted file should be accessible. The status indicator, depending on the application, may be implemented in various ways. For example, the status indicator is stored as a part of the header of the encrypted file. In another example, the status indicator is a separate file that is associated with the encrypted file and contains information related to the decryption process.

[0065] At step 404, decryption requests are sent to members of the authorization group. Depending on the application, these decryption requests may be in various forms, such as an email message, an electronic request, etc. For example, each of the decryption requests includes the link to

the encrypted file and requests an authorization code to be entered. In addition, a decryption request may include detailed information regarding the sender of the requests, that is, who is requesting the access to the requested file. For example, the decryption requests may be sent by a user and/or by a system administrator. According to an embodiment, the number of decryption requests sent is equal to the total number of users in the authorization group.

[0066] At step 405, authorization codes are received. Each of the users who receives a decryption request may provide her or his own authorization code. As explained elsewhere herein, the authorization code for each user may be user specified and/or randomly generated. According to a specific embodiment, the authorization code is provided specifically by each user through a user interface, where the user can provide authorization and, in certain cases, user identification information as well.

[0067] At step 406, the status indicator is updated based on the authorization codes received. In an embodiment, the status indicator records the number of the successful entries of authorization codes. In addition, the status indicator, as explained above, may also include an indication as whether the predetermined threshold number of authorization codes have been received to allow access to the encrypted file and/or decryption thereof.

[0068] At step 407, access to the encrypted file is allowed if various conditions have been satisfied. These conditions include, among other things, the number of successful authorization codes that have been received. As explained above, the number of authorization codes to be received is equal to or less than the total number of authorization codes. In addition, depending on the encryption policy associated with the encrypted file, there may be other conditions as well. For example, the encryption policy may require that the require number of authorization codes include authorization codes of the one or more members of the authorization group. In certain embodiments, the sequence in which authorization codes are received is also a condition which needs to be satisfied before access to the encrypted file is allowed.

[0069] On the other hand, if it is determined that the conditions for granting access are not satisfied, the access is not granted and the process proceeds at step 405 where more authorization codes can be received.

[0070] FIG. 5 is a simplified diagram illustrating an alternative process for access an encrypted file according to an embodiment of the present technology. This diagram is merely an example, which should not unduly limit the scope of the technology or of the appended claims, and includes the following steps:

1. receive an encrypted file (501);
2. provide a decryption status indicator for the encrypted file (502);
3. send decryption request and the encrypted file to users (503);
4. receive authorization codes from users (504);
5. update the status indicator (505);
6. forward the decryption request and the encrypted file to a different user (506); and
7. decrypt the encrypted file if condition satisfied (507).

[0071] At step 501, an encrypted file is received. The considerations discussed elsewhere herein with respect to step 401 apply.

[0072] At step 502, a status indicator associated with the encrypted file is provided. The considerations discussed above with respect to step 402 apply.

[0073] At step 503, decryption requests and the encrypted file are sent to members of the authorization group. According to an embodiment, the encrypted file is sent to a member of the authorization group. For example, the order in which the encrypted file is sent may be based on a predetermined policy and/or arranged by the sender. The encrypted file is sent along with a decryption request. For example, a decryption request may include detailed information regarding the sender of the requests, that is, who is requesting the access to the requested file. For example, the decryption requests may be sent by a user and/or by a system administrator. According to an embodiment, the number of decryption requests sent is equal to the total number of users in the authorization group. According to certain embodiments, to ensure that the correct version of the encrypted file is transferred for the purpose of decryption, a version number (and/or version history and time stamp) is attached to the encrypted file. For example, once sent, the prior copy of the encrypted file may be deleted.

[0074] At step 504, an authorization code for the encrypted file is received from the recipient (i.e., a member of the authorization group). Depending on the application, this may be accomplished in various manners. For example, user partially opens the encrypted file, which prompts the user to

enter a user name and a password thereof. In certain embodiments, the user uses a special application to access the encrypted file and provides the authorization code accordingly.

[0075] At step 505, the status indicator is updated based on the authorization codes received. The considerations discussed above with respect to step 406 apply.

[0076] Based on the decryption status, the encryption file is either forwarded to a different member of the authorization group or decrypted. For example, if the number of corrected authorization codes received is fewer than the predetermined number, the encryption file is forwarded to another member of the authorization group (see step 506). On the other hand, if the number of authorization codes received satisfies the predetermined number, then the decryption process is initiated (see step 507).

[0077] As shown in FIG. 5, at step 506, the encrypted file is forwarded to a different member of the authorization group. For example, the new recipient of the encrypted file is selected based on a predetermined scheme. The choice of forwarding the encrypted file for decryption may be based on a variety of schemes.

[0078] At step 507, the encrypted file is allowed access if various conditions have been satisfied. The considerations discussed above with respect to step 407 apply.

[0079] It is to be appreciated that various embodiments of the present technology provide numerous advantages over other techniques in the art. In a specific embodiment, the present technology provides a flexible method that allows multiple users to determine the access of a protected file, which is useful in many ways, especially in large corporate settings. In addition, certain embodiments of the present technology are compatible with a wide range of existing systems. For example, a system according to present technology may be implemented as a software added on solution. There are other benefits as well.

Specific Implementation of encryption and decryption

[0080] It is to be appreciated that, based on needs, there are alternative specific implementations based on the embodiments of the present technology. According to a specific embodiment, the access policy that requires a predetermined number of users to authorization access to one or more file is implement by software, which verifies and counts user authorization codes and determines

whether to grant access accordingly. According to another embodiment, which is discussed below, user authorization codes are used in the encryption process.

[0081] FIGS. 6 and 7 are simplified diagrams illustrating specific encryption and decryption processes according to embodiments of the present technology. This diagram is merely an example, which should not unduly limit the scope of the technology herein or the appended claims. One of ordinary skill in the art would recognize many variations, alternatives, other embodiments, and modifications of the technology described herein. For example, various steps may be added, removed, replaced, rearranged, repeated, modified, overlapped, and/or partially overlapped, and should not unduly limit the scope of claims. As merely an example, the processes can be implemented in conjunction with the system 100. In addition, these processes may be implemented using a proprietary encryption/decryption software.

[0082] FIG. 6 includes the following steps:

1. receiving one or more plaintext data files (step 601);
2. encrypting the one or more plaintext data files with a first encryption key to generate at least an encrypted file (602);
3. obtaining a number N of encryption codes, each of the encryption codes being associated with a user (step 603);
4. defining an encryption policy, the encryption policy specifying a number M, the number M being a specified number of encryption codes required for performing decryption, wherein M is less than or equal to N and both M and N exceed 1 (step 604);
5. generating N!/(M!*(N-M)!) combination keys, each of the combination keys being a function of M encryption codes and the first encryption key (step 605); and
6. storing the combination keys (606).

[0083] At step 601, one or more plaintext data files are received. It is to be understood that the use of plaintext throughout FIG. 6 is preferred though not required. For example, the data files include information in formats, such as but not limited to plaintext, that are readily accessible to users via various application programs.

[0084] At step 602, the one or more plaintext data files are encrypted using an encryption key. For example, the encryption key can be randomly generated at the time of encryption. In an embodiment, the encryption key is an initialization nonce vector that is used to encrypt the one or more plaintext data files. Depending on the application, other encryption techniques may be used, using the encryption key for the encryption of the one or more plaintext data files. Among other thing, the encryption key is needed for decrypting the one or more plaintext data files.

[0085] At step 603, a number N of encryption codes are obtained. Each of the encryption codes is associated with a user. In a specific embodiment, each of the encryption codes is provided by a user. For example, a user may be prompted to enter a pass code to be used for encryption. As another example, a user may be requested to provide a user specific key file. In an alternative embodiment, encryption codes are randomly generated and then sent to users. For example, each uses receives and stores a randomly generated encryption code.

[0086] According to certain embodiments, each of the encryption codes includes two or more segments. As an example, each of encryption code includes a verification segment and a key segment. The verification segment is used to determine whether an encryption code is correct and to be used for the decryption process. The key segment is used for encrypting and/or decrypting the encryption key. For example, an encryption code is shown below.

Verification	Key Segment
user verification	128 byte key for encryption

Table 1

[0087] During the encryption process, the verification segment is stored at a secured location. During the decryption process, which is described elsewhere herein, users provide encryption codes that contain verification segments, and these verification segments are then verified against the stored verification segments to make sure that the encryption codes from the users are correct.

[0088] At step 604, an encryption policy is defined. Among other things, the encryption policy specifies a number M of encryption codes required for decrypting the encryption key, thereby allowing access to the one or more plaintext data files. As an example, M can be less than or equal to N. Typically both M and N are greater than 1.

[0089] At step 605, a number of combination keys are generated. The number of combination keys is a function of the total number of encryption codes and the required number of encryption codes for access. For example, the total number of combination keys may be equal to N!/(M!*(N-M)!). That is, the total number of combination keys equals to all the possible combinations of M number of encryption codes, where each encryption codes one of N total combination codes. According to embodiments, each of the combination keys is a function of M key segments of encryption codes and the encryption key. For example, M key segments are used together in a reversible manner (i.e., for decoding) to provide encryption for the encryption key.

[0090] Depending on the application, the sequence in which key segments are used in the encryption process may or may not be important during the decryption processes. In a specific embodiment, a specific sequence of key segment used is stored by the encryption software, which performs decryption in a reverse order. In another embodiment, the specific encryption function is used so that the sequence using key segments for encryption and/or decryption is immaterial.

[0091] At step 606, the combination keys are stored. A secured location and special techniques may be used to ensure the security and integrity of the combination keys. For example, the combination keys are stored using a proprietary file structure in a secured location. As another example, the combination keys are stored as a single encrypted file.

[0092] FIG. 7 illustrates a specific decryption process according to an embodiment of the present technology, and includes the following steps:

1. receiving an encrypted file, the encrypted file includes a predetermined number of combination keys, each of the combination keys being a function of a number M of authorization codes (step 701);
2. requesting the authorization codes from a plurality of users, a number N of users having the authorization code, N being greater than M and, typically, both N and M are greater than 1 (step 702);
3. receiving the authorization codes (step 703);
4. verifying the authorization codes (step 704);
5. forming an unlocking key if a number M of authorization codes are received (step 705); and

6. decrypting the encrypted file using the unlocking key (step 706).

[0093] At step 701, an encrypted file is received. According to an embodiment, the encrypted file includes a predetermined number of combination keys, and each of the combination keys is a function of the number M of authorization codes. For example, the encrypted file includes all the combination keys generated during the encryption process (e.g., step 605).

[0094] At step 702, authorization codes are requested from users. For example, there are a number N of users, and requests for authorization codes are sent to each of the N users. In a specific embodiment, a request is sent to users through an electronic message (e.g., e-mail, a text message notification), which informs users as who needed the access, what is being accessed, and that the encryption codes are required.

[0095] The authorization codes can be the same as the encryption codes discussed above. In certain implementations, authorization codes are different from encryption codes. For example, authorization codes include user names and password. By providing authorization codes, the users then may provide encryption codes for decrypting the encryption key.

[0096] At step 703, authorization codes are received by the decrypting software. For example, users can send authorization codes and/or encryption codes through a secured network connection.

[0097] At step 704, the authorization codes are verified. In a specific embodiment, the authorization codes are user names and password that are verified against pre-stored user names and passwords. In another embodiment, the authorization codes that are received are the same as the encryption codes described above, and the verification is performed by checking the verification segment of the encryption codes.

[0098] At step 705, an unlocking key is generated if the number M of encryption codes are received. For example, the unlocking key is the same as the encryption key described above, which is used for both encryption and decryption of plaintext data files. According to an embodiment, the decrypting software gathers the encryption keys (e.g., key segments) and determines whether the M number of codes has been received. If so, the software uses the received encryption codes to determines which of the combination key to decrypt, and uses the M number of available encryption codes to form an unlocking key.

[0099] At step 706, the encrypted file is decrypted using the unlocking key. Depending on the encryption algorithm used, the decryption process may be performed in various ways.

[0100] Although specific embodiments of the present technology have been described, it will be understood by those of skill in the art that there are other embodiments that are equivalent to the described embodiments. Accordingly, it is to be understood that the technology is not to be limited by the specific illustrated embodiments, but only by the scope of the appended claims.

WHAT IS CLAIMED IS:

1. A method for encrypting one or more data files, the method comprising:

specifying an encryption policy for encrypting the one or more data files and generating an output file, the encryption policy including at least an authorization group and a number of authorizations within the authorization group being required for accessing the output file;

defining the authorization group, the authorization group including a plurality of users being able to satisfy the number, each of the plurality of users having an authorization code; and

encrypting the one or more data files in accordance with the encryption policy.

2. The method of claim 1 further comprising compressing the one or more data files.

3. The method of claim 1 further comprising:

generating a plurality of authorization codes; and

providing each of the plurality users with one of the plurality of the authorization codes.

4. The method of claim 1 further comprising:

receiving the authorization code from a user input.

5. The method of claim 1 wherein the number is less than a total number of the plurality of users.

6. The method of claim 1 wherein the encrypting the one or more data files comprises performing an XOR function between the one or more data files and an encryption key.

7. The method of claim 1 further comprising forming a data structure for the output file, the data structure including information associated with the encryption policy.

8. The method of claim 1 further comprising generating an encryption key.

9. The method of claim 1 wherein the output file comprises encrypted one or more data files.

10. The method of claim 1 wherein the output file comprises an encryption key.

11. The method of claim 1 further comprising generating a plurality of authorization codes for the plurality of users.

12. The method of claim 1 further comprising defining the plurality of users.

13. The method of claim 1 wherein the output file is stored at a secure network drive.

14. The method of claim 1 wherein the encryption policy defines an access policy.

15. The method of claim 1 wherein the number is associated with a read level access for the output file.

16. A method for decrypting a file, the method comprising:

receiving a file, the file being encrypted in accordance with an encryption policy, the encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file;

receiving authorization codes from members of the authorization group;

determining a second number, the second number being a number of correct authorization codes received from members of the authorization group; and

decrypting the file if the second number is equal to or greater than the first number.

17. The method of 16 wherein each of the authorization codes is unique.

18. The method of claim 16 wherein the second number of members consists of a subset of members from the authorization group.

19. The method of claim 16 wherein the file comprises an encryption key.

20. The method of claim 16 wherein the second number is greater than one.

21. The method of claim 16 wherein the authorization codes are received non-sequentially.

22. The method of claim 16 further comprising updating a status for the file based on the authorization codes provided.

23. The method of claim 16 wherein:

the file is associated with a decryption key and an encryption key; and

the decryption key and the encryption key are different.

24. A method for decrypting a file, the method comprising:

receiving a file, the file being encrypted in accordance with an encryption policy, the encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file, the authorization group including a plurality of users;

storing the file on a storage, the storage being accessible by the plurality of users;

providing a status indicator associated with a decryption status for the file;

sending a plurality of decryption requests to the plurality of users;

receiving authorization codes from a second number of the plurality of users;

updating the status indicator based on the authorization codes received from the second number of the plurality of users; and

decrypting the file if the second number is equal to or greater than the first number.

25. The method of claim 24 wherein a name associated with the file is hidden.

26. The method of claim 24 further comprising sending a link for the file to the plurality of users.

27. The method of claim 24 further comprising providing a encryption key file if the second number is equal to or greater than the first number.

28. The method of claim 24 further comprising providing instructions associated with the decryption requests.

29. The method of claim 24 wherein the decryption requests comprise email messages.

30. The method of claim 24 wherein the storage comprises a network drive.

31. The method of claim 24 further comprising sending the file to the plurality of users.

32. The method of claim 24 wherein the status indicator is stored at a section of the file.

33. A system for encrypting one or more files, the system comprising:

a storage being configured to store one or more data files;

a user interface, the user interface including a display and an input device; and

a processor;

wherein:

the user interface is configured to receive instruction for specifying an encryption policy for encrypting the one or more data files and generating an output file, the encryption policy including at least an authorization group and a number of authorizations within the authorization group being required for accessing the output file;

the processor is configured for defining the authorization group, the authorization group including a plurality of users being able to satisfy the number, each of the plurality of users having an authorization code;

the processor is further configured for encrypting the one or more data files in accordance with the encryption policy; and

the storage is configured to store the output file.

34. The system of claim 33 wherein the storage is located in a storage area network.

35. The system of claim 33 wherein the storage includes at least one hard drive.

36. The system of claim 33 wherein the number comprises a predetermined threshold number of users required for decrypting the output file.

37. The system of claim 33 wherein:

the output file comprises a data structure, the data structure including a plurality of entries;

the plurality of entries includes a first entry for plurality of encryption keys;

the plurality of entries includes a second entry for a status indicator; and

the plurality of entries includes a third entry for the encryption policy.

38. The system of claim 33 wherein names for the one or more data files are hidden.

39. A method for encrypting plaintext data, the method comprising:

encrypting the data with a first encryption key to generate at least an encrypted file;

obtaining a number N of encryption codes, each of the encryption codes being associated

with a user;

defining an encryption policy, the encryption policy specifying a number M, the number M being a specified number of encryption codes required for performing decryption;

generating at least N!/(M!*(N-M)!) number of combination keys, each of combination keys being a function of M number of encryption codes and the first encryption key; and

storing the combination keys.

40. The method of claim 39 further comprising generating user keys, wherein each of the user keys comprise a verification segment and a key segment.

41. The method of claim 39 wherein each of the combination keys is characterized by a size that is equal to a size of the first encryption key.

42. The method of claim 39 further comprising receiving encryption codes from a plurality of users.

43. The method of claim 39 further comprising storing the encrypted file at a secured location.

44. The method of claim 39 further comprising:

generating the encryption codes; and

sending the encryption codes to a plurality of users.

45. The method of claim 39 wherein the combination keys are stored in a file.

46. The method of claim 39 wherein the combination keys wherein the combination keys are stored at a location different from a location associated with the encrypted file.

47. A method for accessing an encrypted file, the method comprising:

receiving an encrypted file, the encrypted file includes a predetermined number of combination keys, each of the combination keys being a function of a number M of authorization codes;

requesting the authorization codes from a plurality of users chosen from a number N of users having the authorization code, N being greater than M;

receiving the authorization codes;

verifying the authorization codes;

forming an unlocking key if a number of verified authorization codes is equal to or exceeds the number M; and

decrypting the encrypted file using the unlocking key.

48. The method of claim 47 wherein each of the authorization codes comprises:

a verification segment; and

a key segment.

49. The method of claim 48 wherein the verifying the authorization codes comprises matching the verification segment with a pre-stored verification code.

50. The method of claim 48 wherein the unlocking key is a function of key segments.

51. The method of claim 47 wherein the encrypted file comprises a key file for unlocking a plaintext file.

52. The method of claim 47 wherein each of the authorization codes comprises 128 bytes.

53. The method of claim 47 wherein the authorization codes are characterized by a same length.

54. The method of claim 47 wherein the receiving the authorization codes comprises:

receiving requests from users to release authorization codes; and

verifying the requests.

55. The method of claim 47 further comprising verifying the unlocking key.

56. The method of claim 47 wherein the predetermined number of combinations is equal to N!/(M!*(N-M)!).

57. A computer program product for encrypting one or more data files encoded on a computer-readable medium, the computer program product operable to cause data processing apparatus to perform operations comprising:

specifying an encryption policy for encrypting the one or more data files and generating an

output file, the encryption policy including at least an authorization group and a number of authorizations within the authorization group being required for accessing the output file;

defining the authorization group, the authorization group including a plurality of users being able to satisfy the number, each of the plurality of users having an authorization code; and

encrypting the one or more data files in accordance with the encryption policy.

58. A computer program product for decrypting a data file, the computer program product encoded on a computer-readable medium and operable to cause data processing apparatus to perform operations comprising:

receiving a data file, the file being encrypted in accordance with an encryption policy, the encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file,

receiving authorization codes from members of the authorization group;

determining a second number, the second number being a number of correct authorization codes received from members of the authorization group; and

decrypting the file if the second number is equal to or greater than the first number.

59. A computer program product for decrypting a file, the computer program product encoded on a computer readable medium and operable to cause data processing apparatus to perform operations comprising:

receiving a file, the file being encrypted in accordance with an encryption policy, the encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file, the authorization group including a plurality of users,

storing the file on a storage, the storage being accessible by the plurality of users;

providing a status indicator associated with a decryption status for the file;

sending a plurality of decryption requests to the plurality of users;

receiving authorization codes from a second number of the plurality of users;

updating the status indicator based on the authorization codes received from the second number of the plurality of users; and

decrypting the file if the second number is equal to or greater than the first number.

60. A computer program product for encrypting plaintext data, the computer program product encoded on a computer readable medium and operable to cause data processing apparatus to perform operations comprising:

encrypting the data with a first encryption key to generate at least an encrypted file;

obtaining a number N of encryption codes, each of the encryption codes being associated with a user;

defining an encryption policy, the encryption policy specifying a number M, the number M being a specified number of encryption codes required for performing decryption;

generating at least N!/(M!*(N-M)!) number of combination keys, each of combination keys being a function of M number of encryption codes and the first encryption key; and

storing the combination keys.

61. A computer program product for accessing an encrypted file, the computer program product encoded on a computer readable medium and operable to cause data processing apparatus to perform operations comprising:

receiving an encrypted file, the encrypted file including a predetermined number of combination keys, each of the combination keys being a function of a number M of authorization codes;

requesting the authorization codes from a plurality of users chosen from a number N of users having the authorization code, N being greater than M;

receiving the authorization codes;

verifying the authorization codes;

forming an unlocking key if a number of verified authorization codes is equal to or exceeds the number M; and

decrypting the encrypted file using the unlocking key.

62. A system for decrypting a file, the system comprising:

a storage being configured to store one or more files; and

a processor configured to perform operations comprising:

receiving a file, the file being encrypted in accordance with an encryption policy, the encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file;

receiving authorization codes from members of the authorization group;

determining a second number, the second number being a number of correct authorization codes received from members of the authorization group; and

decrypting the file if the second number is equal to or greater than the first number.

63. A system for decrypting a file, the system comprising:

a storage being configured to store one or more files; and

a processor configured to perform operations comprising:

receiving a file, the file being encrypted in accordance with an encryption policy, the encryption policy including at least an authorization group and a first number of authorizations within the authorization group being required for accessing the file;

providing a status indicator associated with a decryption status for the file;

sending a plurality of decryption requests to the plurality of users;

providing receiving authorization codes by from a second number of the plurality of users;

updating the status indicator based on the authorization codes received from the second number of the plurality of users; and

decrypting the file if the second number is equal to or greater than the first number.

64. A system for encrypting plaintext data, the system comprising:

a storage being configured to store one or more files; and

a processor configured to perform operations comprising:

encrypting the data with a first encryption key to generate at least an encrypted file;

obtaining a number N of encryption codes, each of the encryption codes being associated with a user;

defining an encryption policy, the encryption policy specifying a number M, the number M being a specified number of encryption codes required for performing decryption;

generating at leastN!/(M!*(N-M)!) number of combination keys, each of combination keys being a function of M number of encryption codes and the first encryption key; and

storing the combination keys.

65. A system for accessing an encrypted file, the system comprising:

a storage being configured to store one or more files; and

a processor configured to perform operations comprising:

receiving an encrypted file, the encrypted file including a predetermined number of combination keys, each of the combination keys being a function of a number M of authorization codes;

requesting the authorization codes from a plurality of users chosen from a number N of users having the authorization code, N being greater than M;

receiving the authorization codes;

verifying the authorization codes;

forming an unlocking key if a number of verified authorization codes is equal to or exceeds the number M; and

decrypting the encrypted file using the unlocking key.

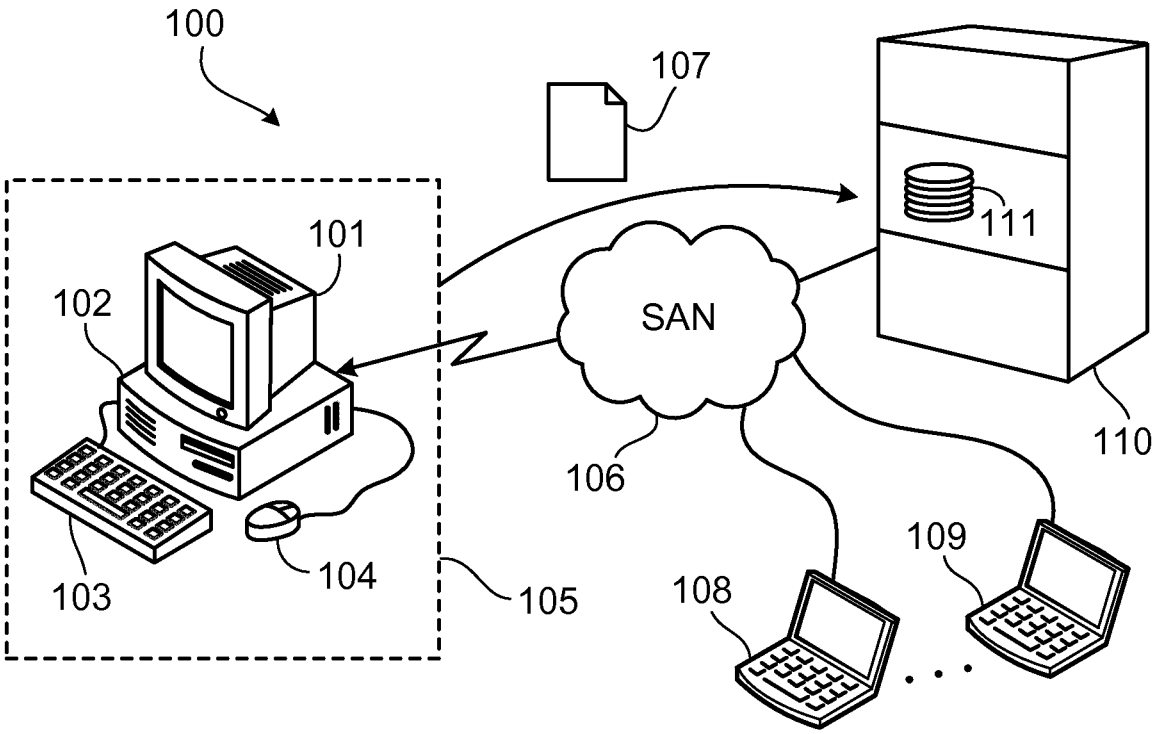

FIG. 1

FILE NAME	DISPLAYED OR HIDDEN
DECRYPTION KEY	MASTER KEY
ACCESS LEVEL/POLICY	K/W/SN? SEQUENTIAL
KEY STATUS	UNLOCKED KEYS? ORDER
VERIFICATION	RANDOM # VERIFY
KEYS	K1, K2, K3, ... Kn
ENCRYPTION METHOD	TYPE?

FIG. 2

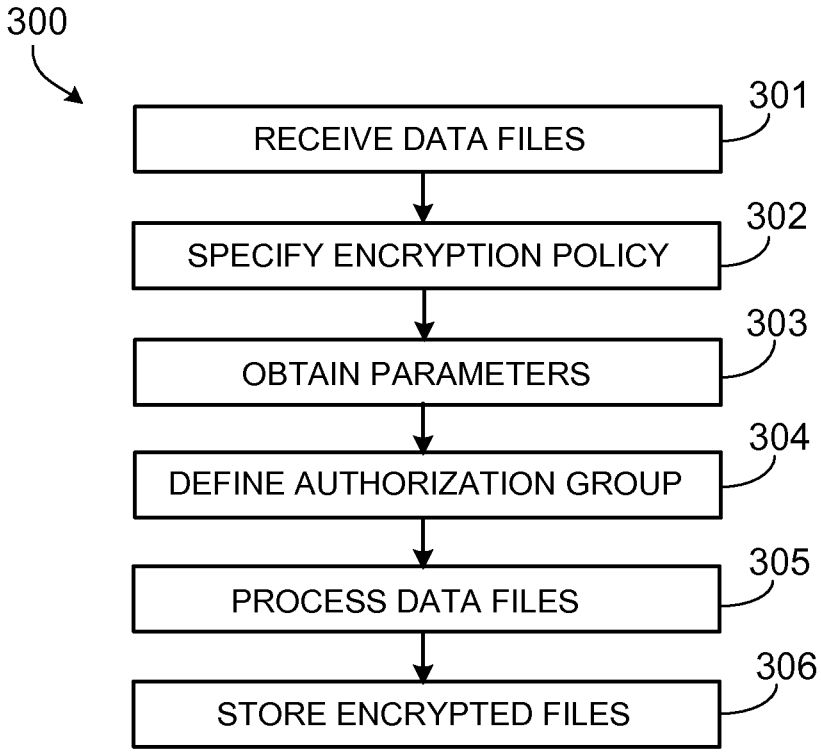
300
RECEIVE DATA FILES
301
SPECIFY ENCRYPTION POLICY
302
OBTAIN PARAMETERS
303
DEFINE AUTHORIZATION GROUP
304
PROCESS DATA FILES
305
STORE ENCRYPTED FILES
306

FIG. 3

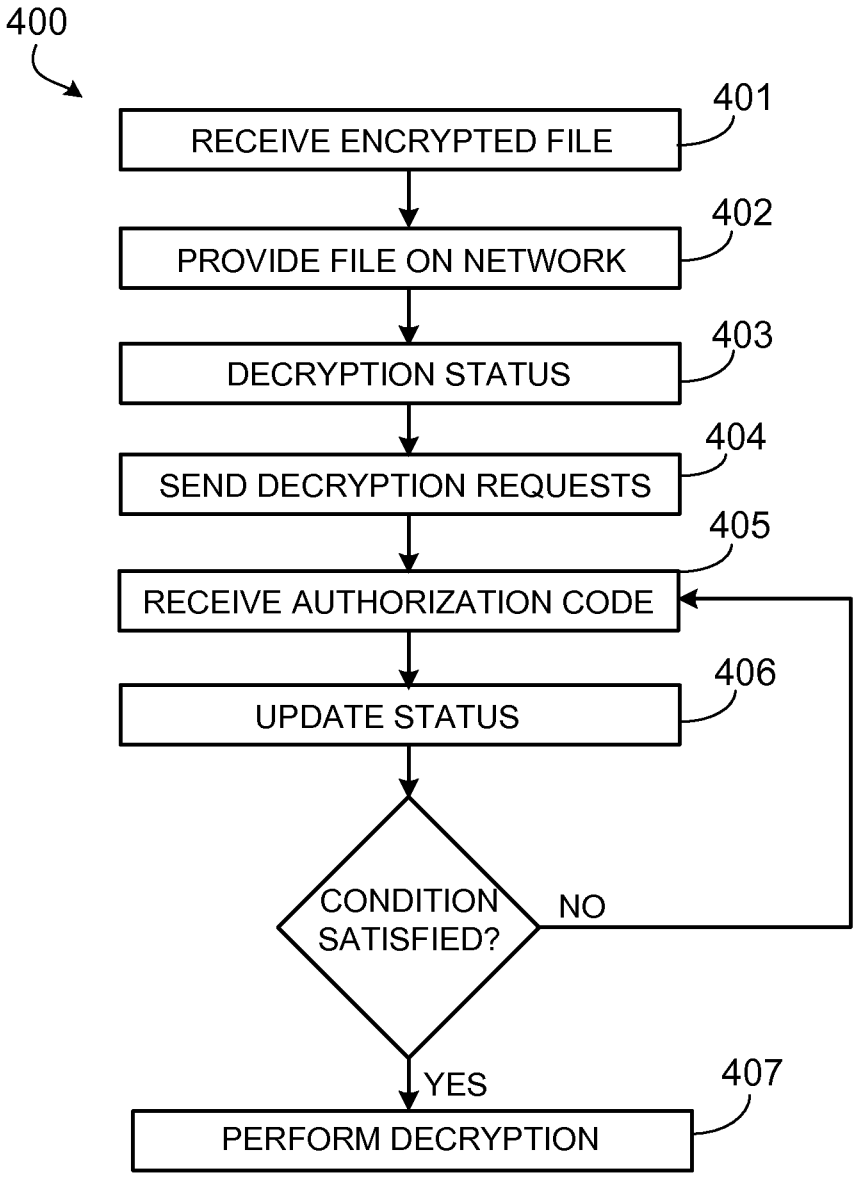
400
401
RECEIVE ENCRYPTED FILE
402
PROVIDE FILE ON NETWORK
403
DECRYPTION STATUS
404
SEND DECRYPTION REQUESTS
405
RECEIVE AUTHORIZATION CODE
406
UPDATE STATUS
CONDITION SATISFIED?
NO
YES
407
PERFORM DECRYPTION

FIG. 4

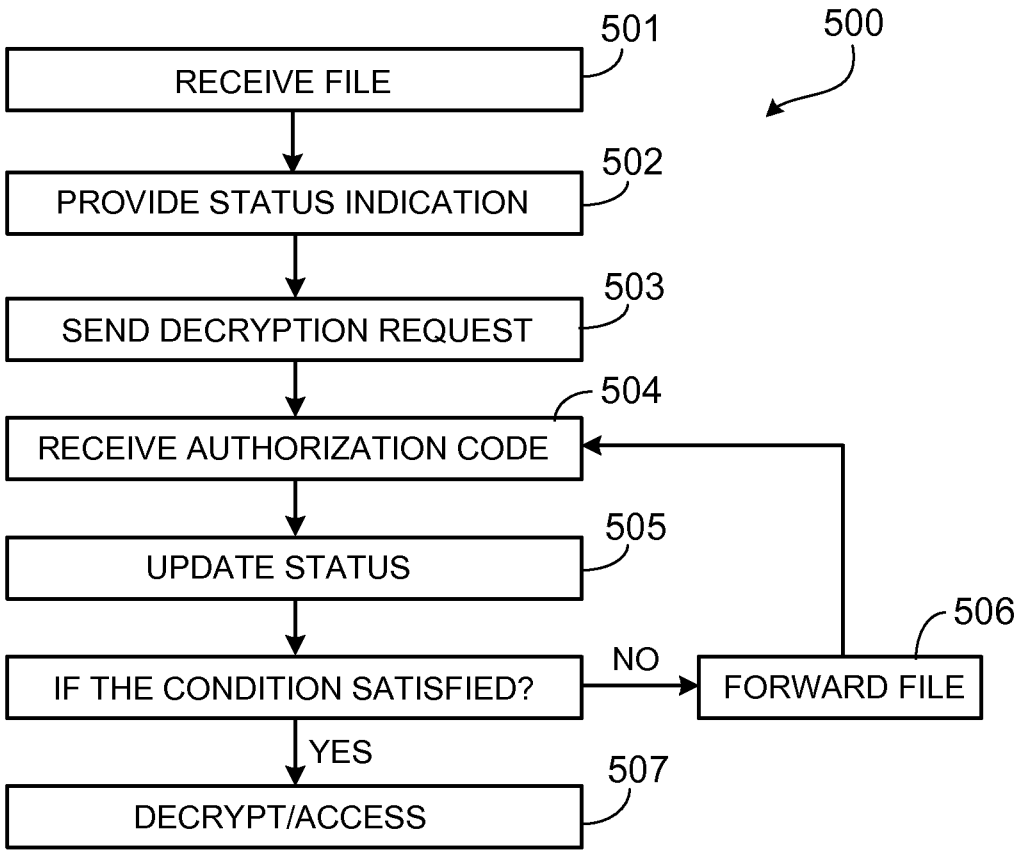
500
501
RECEIVE FILE
502
PROVIDE STATUS INDICATION
503
SEND DECRYPTION REQUEST
504
RECEIVE AUTHORIZATION CODE
505
UPDATE STATUS
506
IF THE CONDITION SATISFIED?
NO
FORWARD FILE
YES
507
DECRYPT/ACCESS

FIG. 5

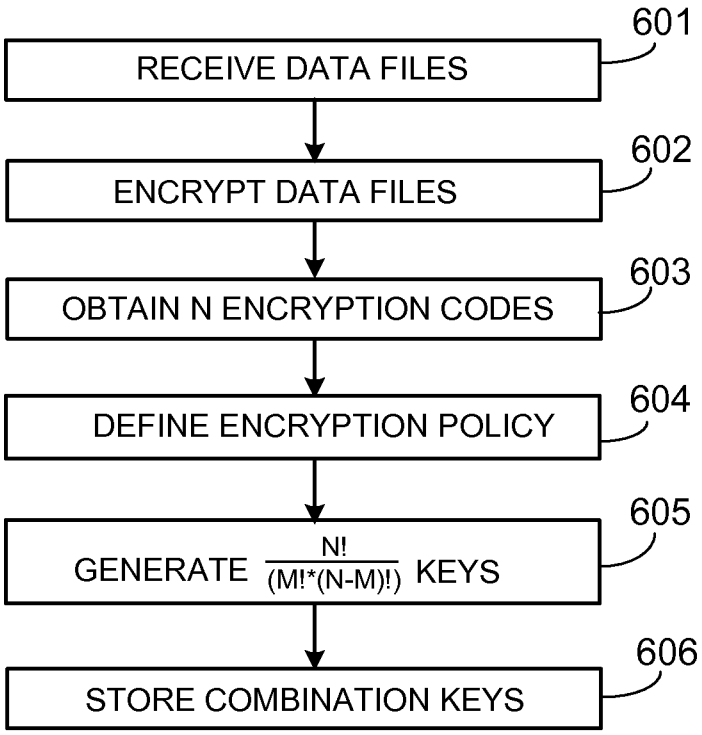
601
RECEIVE DATA FILES
602
ENCRYPT DATA FILES
603
OBTAIN N ENCRYPTION CODES
604
DEFINE ENCRYPTION POLICY
605
GENERATE N! / (M!*(N-M)!) KEYS
606
STORE COMBINATION KEYS

FIG. 6

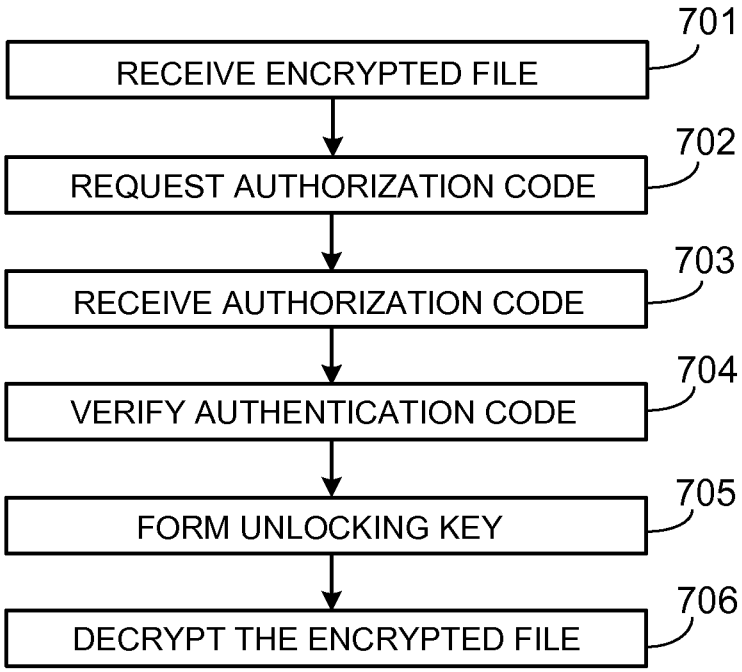
701
RECEIVE ENCRYPTED FILE
702
REQUEST AUTHORIZATION CODE
703
RECEIVE AUTHORIZATION CODE
704
VERIFY AUTHENTICATION CODE
705
FORM UNLOCKING KEY
706
DECRYPT THE ENCRYPTED FILE

FIG. 7