(54) Titre : PROCEDE ET SYSTEME PERMETTANT DE GERER UN PARTAGE DE GROUPE DANS UN STOCKAGE DE DONNEES DISTRIBUE, EN PARTICULIER DANS UN ENVIRONNEMENT P2P
(55) Title: METHOD AND SYSTEM FOR HANDLING OF GROUP SHARING IN A DISTRIBUTED DATA STORAGE, PARTICULARLY IN P2P ENVIRONMENT

(57) Abrégé/Abstract:
Method and system for handling group sharing in distributed data storage environment, to utilize online unexploited storage space and bandwidth of users. Files of a user are cut into pieces which are then distributed among other online users. The original files are stored retrievably. Key Lock Boxes (KLB) are used for storing keys encrypted. With an authentication and key exchange protocol a common group key is generated for encrypting a Master Key Lock Box. The Master KLB represents the root of an oriented tree graph, the nodes of the graph represent KLBs. There is an oriented edge between two nodes when a source KLB (17) contains the key of a target KLB (19). The KLBs (13, 14, 17, 19) contain keys (18, 20) to a subset of files. Starting from said Master Key Lock Box by using the common group key the KLBs (13, 14, 17, 19) are opened until the requested file is reached.
ABSTRACT

METHOD AND SYSTEM FOR HANDLING OF GROUP SHARING IN A DISTRIBUTED OR CLOUD DATA STORAGE ENVIRONMENT

Method and system for handling group sharing in distributed data storage environment. Files are encrypted with unique keys stored in a storage system. Key Lock Boxes (KLB) are used for storing file keys encrypted. With an authentication and key exchange protocol, a common group key is generated for encrypting a Master Key Lock Box. The Master KLB represents the root of an oriented tree graph, the nodes of the graph represent KLBs. There is an oriented edge between two nodes when a source KLB (17) contains the key of a target KLB (19). The KLBs (13, 14, 17, 19) contain keys (18, 20) to a subset of files. Starting from said Master Key Lock Box by using the common group key, the KLBs (13, 14, 17, 19) are opened until the requested file is reached.
METHOD AND SYSTEM FOR HANDLING OF GROUP SHARING IN A DISTRIBUTED OR CLOUD DATA STORAGE ENVIRONMENT

The invention relates to a method and system for handling of group sharing in a distributed or cloud data storage environment. During the method specified files of a given user are stored in an encrypted form. The files can be accessed via their unique identifiers and decrypted using cryptographic keys stored in Key Lock Boxes. The Key Lock Boxes may also be stored as file on the storage system.

Today's personal computers have increased storage capacity as well as faster and more reliable Internet connection. However, a considerable portion of the storage space and the bandwidth remains unutilized. At the same time the possibility for online distribution of contents (e.g. online photo album) including contents edited collectively by a community is a growing demand. This may be for example collecting photos of a certain event using Picasa; collaborative work using Microsoft Groove possibly Google Docs or a version management system, e.g. SVN. If a system is sufficiently reliable then users prefer using this kind of online storage space as online backup, e.g. photographs can be kept synchronous in the local and online storage space by means of Google Picasa. However, Google Storage connected to Google Docs service may also be used as backup storage. Naturally, it is expected that data can be accessed only by users amongst which the data is distributed, i.e. authorized users.

Consequently there is a growing demand for such storage space:
Req1 which can be easily accessed online, i.e. through Internet connection;
Req2 the content of which can be distributed and this content is editable by other users;

Req3 distribution can be controlled and only authorized users are able to access;

4 reliable, i.e. retrieval of the stored data is guaranteed.

The task is to make utilization of an existing storage space, which meets requirements Req1, Req2 and Req3. Req1 is guaranteed by and is the responsibility of the used storage space, whether it be a data warehouse, cloud provider, P2P storage network or any other type of storage.
To ensure Req2 and Req3 by means of a suitable protocol the users should the possibility to share data over the virtually created online storage space the access of which can be easily controlled: where permissions to write and read are dynamically granted for the other users. This protocol must guarantee that these permissions are granted exclusively to these users and even a privileged user (data warehouse administrator) is unable to have an influence on it.

With the present invention we do not intend to introduce a new storage, preferably we lean on existing solutions. Hereinafter these systems will be referred to as low level file systems or (if it is not mistakable) simply file systems. At this level it is expected that on providing a key (e.g. file name) a contiguous byte array (file) returns. In this invention we focus on the following problems:

- If Alice (the individual user) encrypts files so that only her colleagues can read them, how can she send the keys to them?
- How can we guarantee the integrity of the files, that is, how will Alice know that the modifications were made by her colleagues, and not by others?
- If Alice has a new colleague, how can he join the group? Or, if an old one leaves how is he prevented from accessing the files?

These problems must be solved in a higher level layer. In terms of practicability an integrity check should be performed at the time when the files are used.

A high level file system must handle validation of authentication, that is, the already authenticated user must decode the file by using the right key or in case of modification the modified content must be provided with the appropriate integrity protective code.

It is not enough to be able to refer to an object by using a unique identifier, we would like to access the files organized in directory hierarchy so that a unique name can be given to them. A special module, a file level rating unit must ensure that the high level identifier (access path, file name) can be modified.

Identification, authentication of users is essential, to this an additional module is required. The task of this module is to generate a common secret which is known by each of the authorized users but none of the unauthorized ones. This common secret can be used for decoding the keys of the files for reading or the
keys by means of which the modified file can be signed.

On the one hand data must be encrypted in order to prevent unauthorized persons from reading the contents of files with confidential information. Considering that there may be a great amount of data (e.g. a database dump) the use of asymmetric encryption must be minimized as it requires many calculations. Therefore symmetric, effective cryptographic methods based on block encryption should be used.

On the other hand integrity of data must be ensured since storages are untrusted, anybody may be able access their content (e.g. a storage/cloud provider is hacked). Integrity can be ensured by using symmetric (e.g. HMAC) or asymmetric (e.g. digital signature) methods. In case of symmetric method the integrity of a given file can be checked only by a person who owns the key with which the integrity protecting code was generated. In case of the asymmetric method the key needed for the check (public key) is separated from the key needed for generating the integrity protecting code (private key). As it would be useful to check the storage whether the just uploaded data is modified by an authorized user asymmetric integrity protecting method must be used.

The present invention is based on modular architecture. In this approach the different problems are solved in separate modules.

The invention is a method according to the preamble for handling of group sharing in a distributed data storage environment wherein authentication and key exchange protocols are used jointly for generating a common group key with which a highlighted Key Lock Box is encrypted as Master Key Lock Box. The Master Key Lock Box represents the root of an oriented tree graph, the nodes of the graph are further Key Lock Boxes, and there is an oriented edge between two nodes if a source Key Lock Box contains the key of a target Key Lock Box, and in a particular case the Key Lock Boxes may have keys to a subset of the files. During the method starting from the Master Key Lock Box by using the common group key the Key Lock Boxes are opened one after the other downwards in the tree until the key of the requested file is reached.

Features of the invention will now be described with reference to the accompanying drawing in which:
Figure 1 shows the levels of the architecture according to the invention; Figure 2 is an operational chart showing authentication modules and KLB (Key Lock Box) modules.

Figure 3 illustrates the cryptographic scheme of a Master Key Lock Box; and

In the embodiment according to Figure 1 Network Storage Module 1 is responsible for the transfer of user data to and from the storage space. Authentication and Agreement module 3 is built on an authenticated key exchange protocol or a key exchange protocol operating over an authenticated channel, in this manner not all of the old group members 7 are needed to be online at the same time when a new member joins or leaves the group.

File Level Module 2 guarantees read permission using the so called AES256-CBC (or any other symmetric cipher algorithm with other cipher mode) encryption while write permission is guaranteed by digital signature, RSA. Encryption and signature both represent separate sub-modules, they can be replaced, substituted by another. Keys of the individual files are generated by File Level Module 2 through scaling, using a so called hierarchical Key Lock Box architecture, thus when the key of a file is changed the master key is not needed to be changed.

Distributed or peer-to-peer file systems are active topics of research, their development is in progress, it can not be told unambiguously which solution is the best. It is important to note that for the present invention we did not search for file systems but for distributed solutions which make some kind of persistent data store possible which can be used as a file system through abstraction.

According to the present invention the authorized unit of the system is the group: on the one hand it is a set of users who have the permission for the same resources. It may be a one-person group. On the other hand the group includes the files which are accessible for the aforementioned users. One file can belong to only one group, however, read or write permission to a file can be given within a group. However, this restriction does not exclude establishing number of group distributions which is customary in centralized distribution. To this the directory to be distributed is assigned to a common group the members of which are the group
creators. Files of a group are arranged in a customary directory hierarchy i.e. in a rooted tree in which links can be established for access. The root is the root directory of the group distribution. In this manner the group can be considered as a high level file system, together with authorities.

Now the architecture will be described in a top-down, i.e. from higher level to lower level approach. We do not take user interface 7A as a starting point since it is not considered as part of the invention. The system according to the present invention is a layered architecture as it is shown in Figure 1, however, these are not strictly considered as layers. Layer organization simply follows the abstraction levels of modules.

The architecture is provided with a facade 8 which can execute high level operations as “Create new group”, “Join to a group”, “Open a file”, “Write a file” “List the content of a directory” etc.

Facade 8 directly uses File Level Module 2 and Authentication and Agreement Module (AAM) 3: at the beginning it is required to give the ID of the group whose files are to be accessed by user 7 and the authentication data to the group. More particularly: The group identifier must be given to File Level Module 2, then meta data describing the group is downloaded (Group Info): what sort of Authentication and Agreement Module 3 should be used, where can the root directory be found, etc. Authentication data e.g. a password or an RSA private key must be given to Authentication and Agreement Module 3 through facade 8. If the authentication is valid Authentication and Agreement Module 3 returns one or two keys, the Read Master Secret and/or the Write Master Secret. With these Master Secrets the File Level Module 2 can decrypt the respective keys belonging to the files. Then the root directory and all other files of the group can be opened through File Level Module 2. Then the File Level Module 2 provides an interface of hierarchical, high level file system. File Level Module 2 stores encryption or signature keys protected by the Master Secret.

Furthermore, the Authentication and Agreement Module 3 is used for handling group changes: leaving and joining the group, creating a new group. Additionally, Authentication and Agreement Module 3 may provide services to File Level Module 2 which can be accessed by the user, through which a directory or
file can be shared with other groups without admitting the users one by one to the already existing group.

In the middle layer there is a storage divided into three sub-modules, the most important of which is the Network Storage Module 1 as a low level file system. This module assigns a dataset (a low level file) to an ID. This module does not organize the files in hierarchy, this is to be done by File Level Module 2.

Network Storage Module 1 handles storage, searching and reading of low level files. This module also stores the file pieces of others, and manages downloads of others. Furthermore, this module manages integrity of low level files: each of the low level files are signed at the time when it is first stored and the signature public key is given to each of the storage peers together with the file pieces. In this manner modified data of only one authorized user is stored at the storage peers since permission for modification can be checked at these nodes, too. Storage peer is a peer that stores the file, not necessarily a member of the sharing group, but joined to the distributed data store system. A storage peer may also be a data center.

The lower level is the Network Communication Module 5. On the one hand this module supports the usual TCP/IP, IP, UDP/IP protocols, and other communication protocols. In general, Network Storage Module 1 of the medium layer can be joined with Network Communication Module 5 of the lower level.

The optional Credit Counter Module 6 helps Network Storage Module 1, controls resource allocation. The policy is to give preference to only the benevolent community users during allocation of resources. Users that abuse the system (overuse) can be penalized. The overuse of the system can be any one but not limited to: using excessive storage or bandwidth. The architecture defines a higher level “file system” over a network file system, in which safe teamwork is supported. The File Level module 2 is the central module of the system according to the invention, basically it coordinates the whole structure. Therefore it will be described in more detail.

A root and authorizations, keys and all of the users belonging to it represent the group. Any operations associated with the group must start from the Group Info object. The user must “find” this object first. However, finding this object is not
trivial, it is supposed that the user has a group ID. This object describes the authentication and key exchange protocol to be used for joining the group. After successful authentication two of the Authentication and Agreement Modules 3 returns two different keys to the user.

Hereinafter one of the master secrets is referred to as Read Master Key the other is referred to as Write Master Key. Each of the group members must have read permission, but it is sufficient if only a subset of members have write permission. In case of some special system files each of the group members must have write permission, this is solved by originating the write key to these files from the Read Master Key.

In Figure 2 it is shown how a reliable file access and write permission can be given to group members 9 and 12 (e.g. Alice and Bob) by means of authentication modules 10 and 11. Through a TGDH-type key derivation tree 15 Master Secrets to a shared directory 16 are derived from keys in KLB 13, 14.

Other key exchange protocols different from TGDH may also be used.

Handling of the actually used keys are performed by means of Key Lock Boxes (KLB) 13, 14. They contain the encrypted keys. A similar solution can be found in publication titled Group Sharing and Random Access in Cryptographic Storage File systems by Fu, Kevin E (MIT, 1999, Master's thesis). The solution according to this publication does not use hierarchical KLB system and key exchange protocols. It uses a centralized system, not a low level distributed architecture. Among others the aim of the present invention is to eliminate the unfavourable effects of these deficiencies.

There is a highlighted Master Key Lock Box which is opened by the so called Master Secret (this is true for both Read and Write Key Lock Boxes and Master Secret ). Key Lock Boxes 13, 14 store encrypted keys with which the individual files (directories) are encrypted/signed. The reason for using Key Lock Boxes 13, 14 instead of using the group key for each file is that:

a) After every change in the group the group key is also changed. In case of group key change there is no need to re-encrypt the whole file system.

b) Implementing lazy re-encryption is easy.

c) Sharing among groups can be performed in such a way that the common
key of a file is derived from the two keys of the two groups respectively.

Group Info among others contains a pointer to the root directory. Starting from this root directory every file and directory belonging to the group can be accessed like in Unix file systems. Furthermore, similar to Unix file systems every file: the directory and the Key Lock Box are (special) files. At low level the architecture is based on iNodes connected to the data storage part of the file. This iNode is similar to the iNodes used in Unix systems, it describes meta data of files. It has an important role during encryption since this file contains data needed for finding the so called Key Lock Box.

The main difference between UNIX and the present invention is that in UNIX systems the iNodes are used over a block organized storage, while our system operates over a file organized storage. As it will be seen, by performing small changes it can also be used over a block organized storage, but it is unnecessary, as most of the distributed, network storage solutions offer object organization of some type (not necessarily file system).

In order to illustrate the operation the following conventions are introduced:

- High Level File (HLF): a file defined by the invention
- Low Level File (LLF): a coherent object on a storage solution, the size of which may vary and can be referred by an ID, which is not necessarily hierarchically organized.

A file can belong to only one group, however it may be mounted to other groups, too. In this case only a link is established to the file, no other data about the file will be stored at the other group (cross-group link).

The architecture according to the invention is based on a network (possibly distributed) storage. It is supposed that it can store our objects in an organized manner, that is, a variable sized, coherent series of bytes that can be accessed by an ID. A storage of this kind is supposed at the lower level. Hereinafter this level will be referred to as Low Level. A lower level object will be referred to as Low Level File, its ID will be referred to as GUID.

Several low level file systems may be used at the same time, therefore besides GUID the type of the medium always must be given.

Defining of iNodes takes place at inner medium level of File Level Module 2.
The iNode stores meta data of a file and a pointer to the raw data. Considering that in certain systems a file may not exceed a determined size, large files are stored in several low level files which are connected by the iNode. In this manner the upper layers become independent from the restrictions of the low level.

Consequently the iNode may refer to several file parts or another iNode. The latter case is called indirect reference.

It can be seen that in a special case it is feasible that the iNode points to fixed sized blocks, then a system similar to UNIX iNode is obtained, and block organized low level storages can be used.

A lot of metadata can be defined for a file.

A high level file is made up of an iNode and the file parts. This file is accessed by the users, in fact this is a high level interface to the file system. The iNode or file parts can not be accessed by the user directly.

The user can access 3 types of files:

- Normal file for storage of data
- Directory
- Link

The link is a higher layer equivalent to iNode with some restrictions. With a link the user can generate a cross-group link or in-group link. In the background the iNode of a link points to another iNode, thus the link is the only high level file. The iNode of its medium level equivalent does not contain file parts. Naturally, at the end of the link there are file parts as indirect reference. An important restriction is that the link can not be modified, and only existing file (maybe link) can be referred to when it is generated. In this manner circles are avoided.

The directory operates in the usual way: a directory may contain an optional file. In order to map it to the medium level the file parts store a list about the GUID of the iNodes of the files contained in it.

A normal file is a file in which the content of the file parts can be accessed directly by the user. Reasonably, the content of special files is protected from the users by the system. In these files optional data can be stored either in binary or text form.

The method according to the invention defines three special high level files
which are not directly accessible for users:

- Key Lock Box
- Group Info
- System File

Key Lock Boxes are very important for encryption and for ensuring integrity: the Key Lock Boxes store the keys of individual files. In this manner every file comes with a Key Lock Box except for Master (Read and Write) Key Lock Boxes the special feature of which is that it can be opened with the Master Secret and it stores the key of the root directory.

There can be only one Group Info in a group. This is a special file that describes the operation of the whole group. This file may have a lot of additional information, some of them are essential for proper operation:

- The way to get read/write permission, the used authentication protocols and their information
- GUID of the root directory
- GUID of Master Key Lock Boxes

In addition it can have a lot of other information, therefore it is defined in XML format. Higher effectiveness can be achieved with a file of ordered inner format. This is introduced experimentally, considering that handling of data in memory is order of magnitudes faster than accessing files in a distributed storage system, in this manner bottleneck is avoided. With this approach modularity, supplementing can be ensured.

Advantageously the keys used during encryption are stored in Key Lock Boxes. Originally the same key is used for each of the elements of a directory, however, this may change because of lazy re-encryption which will be described in more detail later. Finally in this manner in a Key Lock Box a key can belong to:

- One file
- Several files
- An identification interval
- Every file in a directory

However, in case of many files the size of the Key Lock Box belonging to
the directory may be increased to a large extent which makes handling inefficient. Therefore, in the present invention a hierarchical Key Lock Box 13, 14 structure is used by means of which Key Lock Boxes 13, 14 can be divided to several smaller Key Lock Boxes as required, in order to ensure faster access in the interest of efficiency. This hierarchy is formed similar to B-trees, and it also can be solved that keys of the files used more often be present at higher level. But hierarchy may also be developed in other manner: the individual Key Lock Boxes 13, 14 can be connected to the directories of the file system. In Figure 3 Master Key Lock Box is encrypted by Master Key returned by Authentication and Agreement Module 3 as it was earlier described with reference to Figure 1. In this hierarchy there is a Key Lock Box 19 belonging to Master Key Lock Box 17 if master Key Lock Box contains key 18 to Key Lock Box 19. The hierarchy of Key Lock Boxes 17, 19 can be carried on recursively. Furthermore, each of the Key Lock Boxes 17, 19 have keys 20, 21 for certain files.

Let's suppose that somebody leaves the group of Alice as it was mentioned earlier. In this case the Master Secrets, the keys of the files must be changed and every file must be re-encrypted. This is inefficient, since many of the files would never be changed, only when they are re-encrypted. In a distributed untrusted environment this is much more problematic since after re-encryption all the pieces of the file must be replaced at every peer.

Therefore re-encryption and replacing of online pieces must be avoided in any case because completion of it is practically impossible. A good solution is if a file is re-encrypted only when it is modified, in this manner the number of unnecessary file changes are minimized. This solution does not risk security: it can be supposed that an adversary user has a backup of all the encrypted files. If the file is re-encrypted, this adversary user cannot read its content as he is only aware of the old key.

In the present invention it is implemented in the following way: if e.g. a key 20 is compromised than key 20 is marked with a "dirty" flag in its Key Lock Box 19. If key 20 belongs to a directory or Key Lock Box 19, the dirty flag is recursively applied to every key stored below it in the hierarchy. In this manner through a dirty mark all the keys standing below the Key Lock Box can be disabled, in an extreme
case by marking the keys of the Master Key Lock Box with a dirty flag the keys of all files indirectly become dirty. In case of modification the node which would like to write the file must check whether the used key is dirty. If not, the node can use the file as it was described in the foregoing. If it is dirty, a new key must be generated, stored in the KLB then the whole but already modified file must be stored with the new key. If the key of the KLB is dirty the process is applied recursively.

With respect to write permission much can be lost if the keys of the files are not replaced: if a write key is compromised, an adversary user (this can be an angry, excluded group member) may replace the content of the files by a random number or simply delete them on all peers. Secrecy is not affected but reliability is. However, this is a significant problem in peer-to-peer systems, none of the peer-to-peer systems make efforts to solve it. In the systems under survey the only step taken is that after write permission revocation the write key is replaced one by one for all files. This does not mean that all the files are re-encrypted, so key replacement is much faster, but it is a big task. A race against the adversary user may start: who is faster in replacing the write keys. This quasi equals to lazy algorithm in which the key of the file is replaced only when it would otherwise be modified, since it can be supposed that the adversary user is much faster and is in a more advantageous situation. It is feasible since a user with write permission may behave adversely and delete all files. Thus write permission should only be given to trusted users. In a cloud environment the data centre on which the files are stored on may use a maintained up to date Access Control List (ACL) to prevent an excluded group member from modifying the group’s files.

If ensuring strong integrity for the files of the group is required a difficult problem may be to decide who are the members that can have write permission. What if exactly the user with permission is excluded from the group and the other members are the adversary users. Well, this is an extreme example, in fact generally the majority of the members have the permission. Therefore the majority of the group may decide to whom the permission is given. But how can the storage peers be informed about the decision? Let us take the following example: for managing the affairs of an apartment building in certain offices (storage peers) a person must have an authorization from the majority of owners. Offices can only
check who the owners are. The owners are not able to inform all the offices if there is a change or it takes a lot of time. On the other hand a minority of the owners can not give valid authorization. How can somebody (e.g. a representative) manage the affairs in the name of the owners?

5 a) In a first approach the representative gets the authorization from the majority of the owners. The problem is withdrawal. A solution can be that the authorization is valid only for a certain period of time (e.g. a day). Offices can check the expiry of the authorization (within a certain margin of error). It is a good heuristic, however, withdrawal of authorization can not be surely guaranteed.

10 b) In another approach the majority of the owners go to the office together with the representative and provide evidence of authorization of the representative on the spot, or he is authorized to perform only a particular transaction. This is safe, since authorization can be withdrawn immediately, but it means a large overhead.

15 c) A representative gets a single-use authorization, that is, the certificate of authorization must be handed in and he can not use it again in the same office. However, the representative can copy the certificate (not in the real life, but as it will be seen later, he can do it in an information system), and he can use it in other offices for managing other affairs until a notice is sent to these offices. This may cause inconsistency, furthermore the representative may collect the owners' authorizations and use them at the same time.

In an advantageous embodiment of the invention monitoring distributed storage capacity by a data handling community in a controllable manner becomes possible.

20 KLBs, e.g. Key Lock Box 17 or 19 may also be modelled as an associative array which provides an encryption key for a file ID.

Referring to Figure 2 a generalizable example is shown for handling keys in distributed manner in a KLB structure. The advantage of this is that different files can be encrypted with different keys since encrypting every file with the same key is not practical. On the other hand, due to the KLB structure, re-encryption of the file system is highly effective: instead of re-encrypting every file, only the change is re-encrypted. However, in order to ensure Forward secrecy i.e. to make it sure that
re-encryption will provide future protection, some KLBs must be exchanged upstream in the tree hierarchy towards the root. To this the possibility of using several keys in the file system must be managed. The basic principle itself is well known: the so called lazy re-encryption in which re-encryption is delayed till the time of the first modification, but it has not been realized in a hierarchical system yet.

The hierarchical KLB structure according to the invention can be used for this purpose, too. Figure 2 shows how to derive key words. First Alice is informed through Group Info object which Authentication and Agreement Module (AAM) 3 and what parameters she should use, then sends her own key to the Authentication and Agreement Module 3. After valid authentication the Authentication and Agreement Module 3 returns the proper Master Secret. In case of unauthorized access it is guaranteed mathematically that Alice’s Authentication and Agreement Module 3, 10 will return an incorrect Master Secret, otherwise security can not be guaranteed. The program part implementing Authentication and Agreement Module 10 must be executed by Alice. Using the Master Secret, Master Key Lock Box 13 can be opened in which among others an encrypted key is contained to the KLB of the root directory, which contains the encrypted keys to the root directory and the files stored in it. In addition it also contains a special key to a merged group: In the same manner as Alice could calculate the group key from her own key through her Authentication and Agreement Module 3, any other member of her group can calculate the Group AUB group key by means of the special key contained in the root directory and through an Authentication and Agreement Module 3 even if the keys of other members are changed, in this example it is guaranteed by the merged group key exchange protocol and the TGDH-type key derivation tree 15 implementing Authentication and Agreement Module 3. The same is true for the group of Bob. On the other hand, if Alice opens e.g. a shared directory 16, the keys of the files present in this directory can be found in the KLB belonging to distributed directory 16 in the same manner as the keys of the files present in the root directory. That is, Alice must open the KLB of shared directory 16 to which she has already calculated the key, then she must decipher it and the requested files in shared directory 16 by using this key.

Files may belong to other groups not only a single one. Sharing between
groups can be set in a Windows File Sharing so that members of another group may access certain information. The two groups may change independently of each other, and conveniently, when a member is excluded only from the second group, file sharing is not any more permitted for him and modification of authorization for all resources one by one is not needed for all users. This is ensured by the group merging according to the present invention - this function is provided by means of the KLB structure and key agreement protocol. It operates as follows: a group is created on the basis of key exchange among the users. A group of groups may also be created in such a way that a "representative" (anybody from the group) agrees with the representative of other groups to create a merged group by generating keys, etc. Then the representative stores the content of the agreement in the KLB of his own group. In this manner authentication can be revoked from one of the groups in the same way as from a user in that group. On the other hand, according to the present invention if the group is changed, the parts of the file system must be re-encrypted with new keys. If the group's shared directory were also re-encrypted a member would not be able to access its content since keys can be calculated only within a group. With this method it can be ensured that when a member uses a new key to the shared directory, the members of the other group can calculate this key - this is guaranteed by the key exchange protocol.

An example: Alice's client reads the required data from Group Info object. Authentication and Agreement Module (AAM) 10 returns the Read Master Key. Alice follows the pointer in the Group Info to find the encrypted root directory. The iNode of the root directory has a pointer to the Master Key Lock Box, but Group Info also has this information. Master key Lock Box (MKLB) has a key to the root directory, thus Read MKLB which is opened with Read Master Secret must be read. After decrypting the root directory the KLB belonging to the directory is opened with the key of the directory and can be read. This KLB stores the keys to the files of the given directory and the list contained in it can be read on opening of the directory. When Alice opens a directory in the root directory the procedure must be continued recursively: she decrypts the client directory with the key in the KLB, opens the KLB belonging to the directory, etc.
Implementing of Authentication and Agreement Module 3 is based on TGDH (Tree-based Group Diffie-Hellman) protocol. However, this is not completely satisfactory for using as AAM. For authentication a supplement of it is used: S-TGDH or RSA based authentication. In the embodiment of the present invention broadcast messages are not stored as network messages, they are stored as a special file in a high level file defined by the invention. In this manner it is guaranteed that group members who were not online when the broadcast message was sent, are also informed about the message. This is referred to as Persistent TGDH supplementation. Another TGDH modification is for handling group change: there is always an appointed person (sponsor) who effectively introduces a new member or makes somebody to leave. The protocol does not care if this person is not online when somebody would like to join. In our supplementation in this case a new person is appointed temporarily. Because of mathematic features of the TGDH protocol effectiveness is guaranteed only if joining/leaving of a member is always effected by the originally appointed person. In our supplementation the best sponsor candidate currently available is selected. In order to determine the "goodness" of a sponsor candidate it must be examined to what extent would the optimal height of the TGDH tree be decreased if the given sponsor performed the group operation. Based on the actual position of a candidate in the tree goodness can clearly be defined. Simulations show that in practice guaranteed effectiveness is hardly weakened with this ordered heuristics.

With the solution of the present invention through advantageous resource allocation distributed permission handling can be performed in distributed storage systems.
Claims:

1. Method for handling of group sharing in a distributed data storage environment, in the method specified files of a given user are stored in a manner that they can be retrieved from a used storage system, furthermore Key Lock Boxes are used for storing keys encrypted, characterized in that by using an authentication and key exchange protocol a common group key is generated by means of which a highlighted Key Lock Box as Master Key Lock Box is encrypted, said Master Key Lock Box represents the root of an oriented tree graph, wherein the nodes of the graph are further Key Lock Boxes, and there is an oriented edge between two nodes when a source Key Lock Box (17) contains the key of a target Key Lock Box (19), said Key Lock Boxes (13, 14, 17, 19) may contain keys (18, 20) to a subset of files; during the method starting from said Master Key Lock Box by using said common group key the Key Lock Boxes (13, 14, 17, 19) are opened one after the other downwards in the tree until the key of the requested file is reached.

2. The method according to claim 1 characterized in that the hierarchy of said Key Lock Boxes (13, 14, 17, 19) is assigned to directories present in hierarchy of the directory of the file system.

3. The method according to claim 1 or 2 characterized in that in case of a change the keys of the individual files are re-encrypted by a File Level Module (2) only to the extent of change.

4. The method according to any one of claims 1-3 characterized in that the key exchange protocol is implemented by a TGDH protocol supplemented with persistent operation using the distributed storage system as broadcast channel, further the connecting method according to TGDH protocol is supplemented with a sequence maintaining connection attempt wherein the sequence is determined according to the position occupied laterally or in depth in a TGDH protocol key tree.

5. The method according to any one of claims 1-4 characterized in that said distributed data storage environment is cloud storage environment.

6. System for handling of group sharing in a distributed data storage
environment characterized in that said system comprises a Network Storage Module (1), a File Level Module (2) connected upstream to said Network Storage Module (1) and an Authentication and Agreement Module (3) connected laterally to said File Level Module (2), and a Network Communication Module (5) connected downstream to said Network Storage Module (1).

7. The system according to claim 6 characterized in that a Credit Counter Module (6) is connected laterally to said Network Storage Module (1).

8. The system according to claim 6 or 7 characterized in that said distributed data storage environment is a cloud storage environment.
Fig. 1