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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 August 2016 (18.08.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/128070 A1

(51) International Patent Classification:
G06F 21/62 (2013.01)

(21) International Application Number:
PCT/EP2015/053130

(22) International Filing Date:
13 February 2015 (13.02.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicants: **NEC EUROPE LTD.** [DE/DE]; Kurfürsten-Anlage 36, 69115 Heidelberg (DE). **UNIVERSITÄT MANNHEIM** [DE/DE]; Schloss, 68131 Mannheim (DE).

(72) Inventors: **BOHLI, Jens-Matthias**; Römerstraße 30, 69181 Leimen (DE). **KARAME, Ghassan**; Ringstraße 3, Heidelberg (DE). **ARMKNECHT, Frederik**; Alzeyer Straße 53, 67549 Worms (DE).

(74) Agent: **ULLRICH & NAUMANN**; Schneidmühlstraße 21, 69115 Heidelberg (DE).

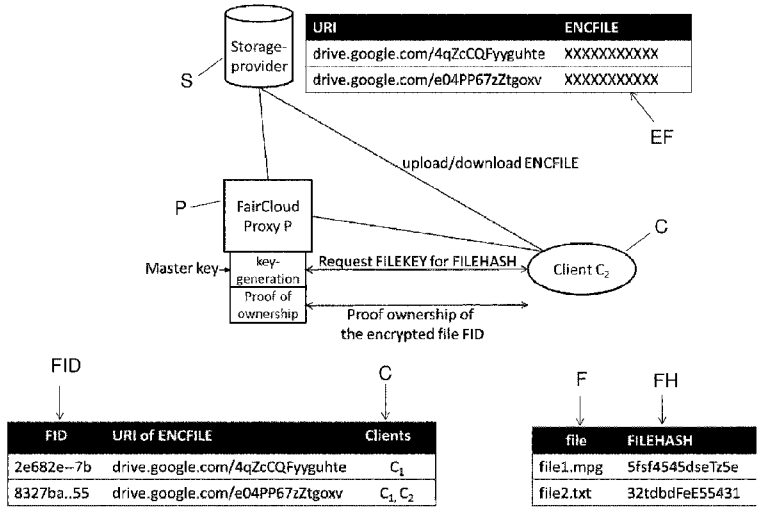
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP,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

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

Published:

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

(54) Title: METHOD FOR STORING A DATA FILE OF A CLIENT ON A STORAGE ENTITY



(57) Abstract: The present invention relates to a method for storing a data file (F) of a client (C) on a storage entity (S), wherein a) a master encryption key is generated by a proxy entity (P) wherein the master encryption key is a deterministic function of the data file (F) to be stored based on a hash value (FH) of a hash-function (H) performed on the data file (F) to be stored by said client (C), b) said data file (F) to be stored is encrypted by the client (C), using the provided master encryption key, c) a hash-tree for the encrypted file (EF) is computed and the top-hash of the computed hash-tree is used as file identification - FID - for the encrypted file (EF), d) the proxy entity (P) checks whether the FID is already known to the storage entity (S) or not and e) in case the FID is not known, the client (C) uploads the encrypted file to the storage entity (S) and to the proxy entity (P), f) the proxy entity (P) computes a top-hash of the encrypted file - PFID - in case the FID is not known or uses a prior computed FID in case the FID is known and performs a proof-of-ownership-procedure for the encrypted data file (F) to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file has been proven, the FID being equal with the PFID is stored on the client together with said hash value (FH).

WO 2016/128070 A1

METHOD FOR STORING A DATA FILE OF A CLIENT ON A STORAGE ENTITY

The present invention relates to a method for storing a data file of a client on a storage entity.

The present invention further relates to a system for storing data of a client on a storage entity comprising a proxy entity.

The present invention even further relates to a proxy entity connectable to a storage entity and a client, preferably for performing with a method according to one of the claims 1-13 and/or a system according to claim 14.

Although applicable to storage in general, the present invention will be described with regard to cloud storage.

Although applicable to any kind of data reducing technique, the present invention will be described with regard to deduplication over encrypted data.

Cloud storage is receiving increasing attention and importance recently. Cloud storage offers their users cost-effective, convenient and highly available storage services. Conventional clouds rely on cost-effective techniques such as data compression and data deduplication in order to save storage costs for the cloud.

Data deduplication clearly comes at odds with data confidentiality. That is, existing semantically secure encryption techniques render any two identical chunks of data indistinguishable to the cloud storage provider, thus preventing the cloud storage provider from effectively deduplicating data.

In the non-patent literature of

- Pasquale Puzio, Refik Molva, Melek Önen and Sergio Loureiro. ClouDedup: Secure Deduplication with Encrypted Data for Cloud Storage, Proceedings of IEEE CloudCom 2013,

- A Secure Data Deduplication Scheme for Cloud Storage, Jan Stanek, Alessandro Sorniotti, Elli Androulaki, and Lukas Kenc, Proceedings of Financial Cryptography and Data Security, 2014,
- Boosting Efficiency and Security in Proof of Ownership for Deduplication, Roberto Di Pietro, Alessandro Sorniotti, Proceedings of ASIACCS 2012, and
- Mihir Bellare and Sriram Keelveedhi, Thomas Ristenpart, DupLESS: Server-Aided Encryption for Deduplicated Storage, Proceedings of Usenix Security 2013,

techniques are disclosed for performing deduplication over encrypted data or for a construction for a proof of ownership to attest that a user indeed possesses a file which is deduplicated by a cloud for example. These conventional techniques do not efficiently protect against malicious users to abuse the system, e.g., upload data encrypted with the wrong encryption key, etc.

However one of the disadvantages is, that these techniques are not transparent for the users of a cloud storage provider. Another disadvantage is, that the users do not have a fine-grained control over their possibly deduplicated files.

It is therefore an objective of the present invention to provide a method and a system for storing a data file of a client on a storage entity which support strong confidentiality, resistance against malicious users who might e.g., wrongfully acquire file hashes, and provide a fine-grained access control over their files.

It is a further objective of the present invention to provide a method and a system for storing a data file of a client on a storage entity which are easy to implement.

It is an even further objective of the present invention to provide a method and a system for storing a data file of a client on a storage entity enabling a scaling with the number of users, file sizes, etc. and without deteriorating the performance witnessed by users and compared to conventional methods and systems.

The aforementioned objectives are accomplished by a method of claim 1, a system of claim 14 and a proxy entity of claim 15.

In claim 1 a method for storing a data file of a client on a storage entity is defined.

According to claim 1 the method is characterized in that

a) a master encryption key is generated by a proxy entity wherein the master encryption key is a deterministic function of the data file to be stored based on a hash value of a hash-function performed on the data file to be stored by said client,

b) said data file to be stored is encrypted by the client, using the provided master encryption key,

c) a hash-tree for the encrypted file is computed and the top-hash of the computed hash-tree is used as file identification – FID - for the encrypted file,

d) the proxy entity checks whether the FID is already known to the storage entity or not and

e) in case the FID is not known, the client uploads the encrypted file to the storage entity and to the proxy entity,

f) the proxy entity computes a top-hash of the encrypted file – PFID – in case the FID is not known or uses a prior computed FID in case the FID is known and performs a proof-of-ownership-procedure for the encrypted data file to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file has been proven, the FID being equal with the PFID is stored on the client together with said hash value.

In claim 14 a system for storing data of a client on a storage entity, comprising a proxy entity is defined.

According to claim 14 the system is characterized by

said proxy entity is adapted to generate wherein the master encryption key is a deterministic function of the data file to be stored based on a hash value of a hash-function performed on the data file to be stored by said client,

said client is adapted to encrypt said data file to be stored using the provided master encryption key, and to compute a hash-tree for the encrypted file and

said proxy entity is adapted to receive the top-hash of the computed hash-tree as file identification – FID - for the encrypted file, to check whether the FID is already known to the storage entity or not,
in case the FID is not known, the client is adapted to upload the encrypted file to the storage entity and to the proxy entity, and wherein
the proxy entity is adapted to compute a top-hash of the encrypted file – PFID – in case the FID is not known or uses a prior computed FID in case the FID is known and performs a proof-of-ownership-procedure for the encrypted data file to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file has been proven, the FID being equal with the PFID is stored on the client together with said hash value.

In claim 15 a proxy entity is connectable to a storage entity and a client, preferably for performing with a method according to one of the claims 1-13 and/or a system according to claim 14 is defined.

According to claim 15 the proxy entity is characterized in that
said proxy entity adapted
to generate a master encryption key wherein the master encryption key is a deterministic function of the data file to be stored based on a hash value of a hash-function performed on the data file to be stored by said client,
to receive a top-hash of a computed hash-tree as file identification – FID - for the encrypted file,
to check whether the FID is already known to the storage entity or not and
in case the FID is not known, to receive the encrypted file from the client,
to compute a top-hash of the encrypted file – PFID – in case the FID is not known or to use a prior computed FID in case the FID is known and
to perform a proof-of-ownership-procedure for the data file to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file has been proven,
to indicate to the client that the FID being equal with the PFID.

According to the invention it has been recognized that storage space when storing encrypted files can be reduced by client-driven deduplication of files.

According to the invention it has been further recognized that users are enabled to fairly and securely share the savings of data deduplication in spite of a rational service provider which might not accurately report the deduplication patterns of the stored data.

According to the invention it has been even further recognized that strong confidentiality is guaranteed and protection against malicious users is enabled who might be interested in abusing a deduplication service.

According to the invention it has been even further recognized that an indexing of files based on a top hash of the encrypted file is enabled with a key derived from a deterministic function ensuring that users cannot cheat by uploading files/their data not correctly encrypted.

According to the invention it has been even further recognized that proof of ownerships can be easily performed by the proxy entity simply by checking that the proof of ownership matches the file identifier FID.

According to the invention it has been even further recognized that an easy implementation is enabled since existing application programming interfaces API can be used provided by conventional service providers.

According to the invention it has been even further recognized that an overhead incurred on the proxy entity for example in orchestrating data deduplication is minimized.

Further features, advantages and preferred embodiments are described in the following sub claims.

According to a preferred embodiment for performing step a) said client blinds said hash-value with an oblivious pseudo-random-function prior to transmitting the blinded value to the proxy entity. This enables in a fast and efficient way to blind the hash-value, thus hading the hash value from the proxy entity.

According to a further preferred embodiment after receiving the blinded value, the proxy entity signs the blinded hash-value and returns it to the client wherein the client then unblinds the signed value, performs the hash-function on the unblinded received value and uses the result as encryption key. This enables in an easy way providing a server-aided/proxy-aided key generation protocol between the client and the proxy entity. Since the master encryption key is a deterministic function of the data file to be stored the master encryption key is "bounded" to the data file and can be efficiently used when the deduplicating data files to be stored.

According to a further preferred embodiment the top hash is computed for a Merkle hash tree or a tiger hash tree. If the hash tree is the Merkle hash tree then a fast and efficient computing of the hash tree and therefore of the top hash can be performed since the Merkle hash tree is an example for a binary hash tree. Alternatively the tiger hash tree can be used using the crypto hash function "tiger". A tiger hash tree hashes for example on the level of the leaves data blocks of a data file each having 1024 bytes.

According to a further preferred embodiment upon a request of the client to the proxy entity to store the file on said storing entity, the proxy entity provides upload information to the client, wherein said upload information is only temporary valid. For example the proxy entity can issue a timed generate URL command enabling the client to upload the data onto its account within a certain time interval. A timed generateURL command results in a URL expiring after a specified period of time. This enables the proxy entity to recognize file uploads and to organize them without having to wait too long until the client uploads the file. If the client does not use the corresponding timed uploading information, then the client – when again trying to upload the file – has to reissue a corresponding request to the proxy entity.

According to a further preferred embodiment upon successful proof-of-ownership

6a) In case the FID is not known, the FID and meta-data associated with the encrypted file is stored by the proxy entity, preferably including client information and the size of the encrypted file,

6b) In case the FID is known, client information is added to the meta-data associated with the stored encrypted data file with corresponding FID.

This enables in an easy and efficient way for example a later download of the data file or deletion of a data file using the meta-data associated with the former uploaded file.

According to a further preferred embodiment in case of 6b) the client deletes the local copy of the data file upon receiving information about successful proof-of-ownership. This saves resources on the client's side since when the client deletes the local copy of the data file, the client has only to store the FID and the original hash of the data file for later manipulation of the data file, for example downloading it again, deleting the data file or the like.

According to a further preferred embodiment for downloading a data file from the storage entity, the client submits the FID to the proxy entity and the proxy entity provides after successful check that the client information of the client matches to the meta-data associated to the data file with said FID, server download information to the client, preferably wherein the server download information are only temporary valid. This enables in an easy and efficient way a download of the data file requested by the clients preferably the proxy entity may note the number of download requests performed by each client for each file.

According to a further preferred embodiment for decrypting the downloaded data file with a decryption key, the client either uses a corresponding cached decryption key associated with the FID or the client performs step a) to acquire the corresponding decryption key. This enables in a flexible way to provide the client with the decryption key to decrypt the encrypted downloaded file. If the client has not stored the corresponding decryption key when having uploaded the data file, the client can request the corresponding decryption key using the master encryption key from the proxy entity again.

According to a further preferred embodiment in case the PFID does not match the FID the data file corresponding to the PFID is deleted from the storage entity. This

ensures that only the files which have been requested by the client for storage are stored at the storage entity, e.g. a cloud storage, a server or the like. A misuse of the storage entity is therefore avoided, at least reduced.

According to a further preferred embodiment the directory operations on a file system of the client are performed locally on the client hidden from the proxy entity. When the client has a file system on which he operates directory operations involving the stored data file are hidden from the proxy entity and thus security of the client is enhanced. Directory operations are preferably comprising directory creation, directory renaming, etc.

According to a further preferred embodiment a data file is stored on the storage entity under a random identifier mapped to the FID. This further enhances the security, since random identifiers cannot be guessed by a client in order to download the file. Further the flexibility is enhanced, since random identifiers can be generated according to the needs of the storage entity for example.

According to a further preferred embodiment when a data file is indicated by a client to be deleted, the proxy entity renames the data file to another random identifier and provides upon a request for access to a renamed data file by another client, corresponding new access information associated to the FID and the renamed data file. This enables in an easy way to delete files in particular in connection with storage providers not supporting URL commands for file creation e.g. only provide non-timed URL-based file download. When a user requests to delete a file, the proxy for example manually renames the data file to another random and unpredictable identifier for said data file. Other legitimate clients who require access to said data file then contact the proxy entity again who informs them of for example a new URL corresponding to the renamed data file.

There are several ways how to design and further develop the teaching of the present invention in an advantageous way. To this end it is to be referred to the patent claims subordinate to patent claim 1 on the one hand and to the following explanation of preferred embodiments of the invention by way of example, illustrated by the figure on the other hand. In connection with the explanation of the

preferred embodiments of the invention by the aid of the figure, generally preferred embodiments and further developments of the teaching will be explained.

In the drawings the only

Fig. shows a system according to an embodiment of the present invention.

A number of clients C1, C2, ... - in Fig. 1 only C2 is shown – are interested in storing their files at a storage provider S. Preferably the storage provider S exposes to its clients C1, C2, ... a standard interface providing a plurality of simple operations such as storing a file, retrieving a file, deleting a file, generating a URL for sending HTTP commands for storage/retrieval, etc.. In case the storage provider S is a commodity cloud service provider he may expose an even more simpler interface for example that does not allow storing a file using a URL.

Clients C1, C2, ... are interested in storing their files at low cost. In the Fig. is shown a gateway or proxy entity P which owns an account hosted by the storage provider S and performs cross-user file-based deduplication of files. It is also possible that users, i.e. clients C1, C2, ... coordinate their file uploads to the storage provider S prior to storing their data on the storage provider S. Such a decentralized coordination however requires interaction among the users respectively clients and is unlikely to scale as a number of users/clients storing the same data increases.

The gateway P is preferably a logically centralized entity and can be easily instantiated using a any number of distributed servers for example. Similar to conventional operations of existing cloud storage providers, the storage provider S for example charges the proxy entity P according to the total storage space that the proxy entity P and the clients C1, C2, ... are consuming and the total number of bytes that they download. In turn the gateway P charges the clients C1, C2, ... according to the data that they are respectively storing after the data has undergone deduplication and to the total number of bytes each client C1, C2, ... has downloaded.

Further it is assumed that the clients C1, C2, ... and the gateway P share user keys and credentials, for example certificates or the like. In particular all communication between a client C1, C2, ... and the gateway P is authenticated and preferably encrypted. It is also assumed that a secure encryption procedure ENC and a cryptographic hash function H is provided.

There are number of problems which arise:

- Access control: Since the gateway P owns the account on the storage entity S the gateway P needs to effectively manage access rights of users/clients onto the stored files. This preferably includes granting/revoking read rights for users on the files and accurately measuring user access patterns for billing purposes later on. The gateway P also needs to ensure that only legitimate users/clients who own and have subscribed to a given file can access the file.

- Malicious users: The gateway P additionally needs to prevent abuse by malicious users who can deviate from the protocol and for example upload ill-constructed content, encrypt their data using unknown keys to prevent honest users from later on accessing their files, etc.. This process has to be efficient for example users cannot constantly download and decrypt uploaded files in order to check their correctness.

- Curious gateway P and storage provider S: Even further both the gateway P and the storage provider S should not be able to acquire any information about the contents of the file stored on the storage provider Seven if they collude and put their information together. Of course both the gateway P and the storage provider S have to know the file sizes and the download patterns of the files in order to perform a correct accounting.

The system shown in Fig. 1 takes the above into account and performs the following steps:

When a client wishes to a upload a new file fi onto the storage entity S, the client Ci issues an upload request to the proxy entity P. Subsequently, the client Ci and the proxy entity P start executing the server-aided key generation protocol. More specifically, the client Ci blinds H(fi) with r^e, where r is a random number, and e denotes the public key of P and d denotes the private key. Upon reception of $H(fi)r^e$, the proxy entity P signs it and returns the signature $H(fi)^dr$ to the client C; the latter unblinds it and computes the key $K=H(H(fi)^d)$. This procedure is not bound to a particular Oblivious PRF protocol and can rely on other protocols which offer similar guarantees. For example, it can be instantiated using blind RSA or blind BLS signatures.

The client Ci then encrypts the file fi using an encryption algorithm Enc under a key K, computes and sends to the gateway P the Merkle root of the Merkle tree over the encrypted file, FID = MTEnc(K; fi). Subsequently, the gateway P checks if any other client has previously stored FID. If the

- FID has not been stored before: In this case, the gateway P issues a timed generateURL command allowing the client Ci to upload the data file onto its account within a time interval d, e.g. using a timed generateURL command resulting in a URL which expires after the specified period of time. After the upload of the encrypted data file Enc(K; fi) terminates, the gateway P accesses the storage provider S and computes the Merkle root of the stored file, and verifies that it matches FID. If the verification matches, the gateway P stores metadata associated with the uploaded file in a newly generated structure indexed by FID: this includes preferably the ID of the client Ci, the size of the underlying file Enc(K; fi) (in bytes). Otherwise, if the Merkle root of the stored file does not match FID, the gateway P deletes the data file and adds the ID of the client Ci to a blacklist (e.g., the gateway P can ban further requests from client Ci).

- FID has been stored before: In this case, the gateway P requests that the client Ci proves that it owns the file FID.
  For that purpose, the gateway P and the client Ci execute a proof-of-work PoW protocol. In essence, the gateway P chooses a random number u of

leaf indexes of the Merkle tree of encrypted file Enc(K; fi), and asks the client Ci for the sibling paths of all the u leaves; the gateway P accepts if all the sibling paths are valid with respect to the Merkle root FID. If this PoW verification passes, the gateway P appends the ID of the client Ci to the file metadata structure with index FID, and sends an ACK to the client Ci. In turn, the client Ci deletes the local copy of the file and only needs to store FID and the original hash of the file H(fi).

To download a file with index FID, a client Ci submits a corresponding FID to the gateway P; the latter checks that the client Ci is a member of the user list added to the metadata structure of FID. If so, the gateway P generates a timed URL allowing the client Ci to download the requested file from the storage provider S. Additionally it is preferably assumed that the gateway P notes the number of download requests performed by each client for each file. If the client Ci did not cache the decryption key associated with FID, then the client Ci can use H(fi) to acquire the corresponding key by executing the server-aided/proxy-aided generation protocol with the gateway P.

When a client Ci wants to delete a file with identification FID, it informs the gateway P. The gateway P marks the client Ci for deletion from FID in the subsequent epoch.

Preferably the clients directory structures are hidden from the gateway P by working on a single directory structure hosted within the storage providers account on the cloud. This has the benefit of reducing the overhead beared by the gateway P, i.e. no path related overhead, but relies on the clients C1, C2, ... storing their directory structure locally and for example storing their encrypted directory structure at the gateway P. Directory operations such as directory creation, directory renaming, etc. are locally maintained by the software client of the users. Local directories comprise pointers to the client files outsourced to the cloud, which enable the local client to perform operations such as directory listing and file renaming without the need to contact the gateway P, thereby minimizing the overhead incurred on the gateway P. Only operations that affect the client files stored on the cloud (e.g., filename search, file deletion/creation) are transmitted to

the gateway P. By hiding the directory structure from the gateway P the interactions with the gateway P and the clients C1, C2, ... are minimized enabling maximum user privacy since the directory structures may leak considerable information about the files stored therein and consequently about the underlying user profile. Preferably the directory structure particular to each user/client C1, C2, ... is stored encrypted at the gateway P thus enabling users to synchronize their directories across multiple devices.

When a cloud service provider S does not support URL commands for file creation for example and only provide non-timed URL-based file download then preferably an URL-based PUT is replaced by the clients C1, C2, ... uploading the data file to the gateway P which in turn uploads the file to the storage provider S. Since the gateway P has to compute the Merkle tree over the uploaded file this is preferably performed before the gateway P uploads the file to the storage provider S therefore reducing the performance penalty incurred on the gateway P.

The files can also be stored on random identifiers and can be accessed by means of permanent URLs which map to the corresponding FID. When the user/client requests to delete a file, the gateway P has to manually rename the file to another random and unpredictable identifier. Other legitimate clients C1, C2, ... requiring access to the file have to contact the gateway P who informs them of the new URL corresponding to the renamed file object.

The present invention enables fine-grained access control on shared files preferably relying on the notion of self-expiring URL when accessing content. Whenever a user wishes to access a given resource the gateway generates the URL for that resource on the fly which expires after the period of time.

Further the present invention enables an easy implementation since conventional cloud application programming interfaces support dynamic generation of such expiring resources URLs.

Even further the present invention does not only ensure that the gateway can restrict access to the data stored on the cloud but enables the gateway to keep track of the access pattern of its users for example to be used in billing later.

Even further the present invention provides an oblivious server-aided or gateway-aided encryption key generation to ensure that the stored files are encrypted with keys that are dependent on both the hash of the file and the gateways secret. This enhances the security against brute force search attacks when the message content is predictable, also ensuring that a curious gateway/storage provider which does not know the file hash cannot acquire the necessary keys to decrypt them.

Even further the present invention provides a proof of ownership over the encrypted file to protect against malicious users who otherwise have obtained the file hash, for example by theft or malware but do not posses the full file. Besides proving that a given users indeed in possession of the full file this guarantees to a user that the cloud stores a file which is correctly encrypted.

The present invention also provides an indexing of files based on the Merkle root of the encrypted file with the key derived from an oblivious pseudo-random function protocol. This ensures that users cannot cheat by uploading files that are not correctly encrypted and that a proof of ownerships can be easily performed by the proxy/gateway simply by checking that the proof matches the file identifier.

The present invention preferably provides a system comprising the steps of

1) Executing a server-aided key generation protocol between users and the proxy to output encryption key, preferably using an oblivious pseudo-random function,
2) Encrypting by the user the file with the obtained encryption key and computing the Merkle root of the encrypted file as FID,
3) If FID is not yet known, the encrypted file is uploaded by the user. The proxy obtains the encrypted file, and also computes a Merkle root and data needed for a proof of ownership,

4) If the file FID is already in the Server/storage entity, the proxy knows data to perform a proof of ownership of the encrypted file with the user.

The present invention has inter alia the following advantages: It efficiently enforces fine-grained access control over deduplicated files, supports strong confidentiality, resistance against malicious users and protects from a rational gateway which attempts to overcharge the users. Further the present invention provides cheaper storage costs than conventional commodity storage servers without compromising the confidentiality of the data or the performance of the system. For example the present invention incurs considerable storage cost savings on cloud users of 30% compared to conventional commodity storage services for a number of realistic profiles of users.

Even further the present invention is transparent from the perspective of the users and the storage provider. The present invention could be implemented within existing application programming interfaces API provided by conventional service providers without deteriorating the performance witnessed by users when compared this conventional solutions where users directly interface with this storage provider. The present invention scales with the number of users, the file size and the number of uploaded files. In particular the overhead incurred on the gateway P in orchestrating data deduplication is minimal while incurring tolerable overhead on users when verifying for example their bills at the end of every time epoch.

Many modifications and other embodiments of the invention set forth herein will come to mind to the one skilled in the art to which the invention pertains having the benefit of the teachings presented in the foregoing description and the associated drawings. Therefore, it is to be understood that the invention is not to be limited to the specific embodiments disclosed and that modifications and other embodiments are intended to be included within the scope of the appended claims. Although specific terms are employed herein, they are used in a generic and descriptive sense only and not for purposes of limitation.

Claims

1. A method for storing a data file (F) of a client (C) on a storage entity (S), characterized in that
   a) a master encryption key is generated by a proxy entity (P) wherein the master encryption key is a deterministic function of the data file (F) to be stored based on a hash value (FH) of a hash-function (H) performed on the data file (F) to be stored by said client (C),
   b) said data file (F) to be stored is encrypted by the client (C), using the provided master encryption key,
   c) a hash-tree for the encrypted file (EF) is computed and the top-hash of the computed hash-tree is used as file identification – FID - for the encrypted file (EF),
   d) the proxy entity (P) checks whether the FID is already known to the storage entity (S) or not and
   e) in case the FID is not known, the client (C) uploads the encrypted file to the storage entity (S) and to the proxy entity (P),
   f) the proxy entity (P) computes a top-hash of the encrypted file – PFID – in case the FID is not known or uses a prior computed FID in case the FID is known and performs a proof-of-ownership-procedure for the encrypted data file (F) to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file has been proven, the FID being equal with the PFID is stored on the client together with said hash value (FH).

2. The method according to claim 1, characterized in that for performing step a) said client (C) blinds said hash-value (FH) with a oblivious pseudo-random-function prior to transmitting the blinded value to the proxy entity (P).

3. The method according to claim 2, characterized in that after receiving the blinded value, the proxy entity (P) signs the blinded hash-value and returns it to the client, wherein the client (C) then unblinds the signed value, performs the hash-function on the unblinded received value and uses the result as encryption key.

4. The method according to one of the claims 1-3, characterized in that the top-hash is computed for a Merkle hash tree or tiger hash tree.

5. The method according to one of the claims 1-4, characterized in that upon request of the client (C) to the proxy entity (P) to store the data file (F) on said storage entity (S), the proxy entity (P) provides upload information to the client, wherein said upload information is only temporary valid.

6. The method according to one of the claims 1-5, characterized in that upon successful proof-of-ownership

6a) In case the FID is not known, the FID and meta-data associated with the encrypted file is stored by the proxy entity (P), preferably including client information and the size of the encrypted file,

6b) In case the FID is known, client information is added to the meta-data associated with the stored encrypted data file (EF) with corresponding FID.

7. The method according to claim 6, characterized in that in case of 6b) the client (C) deletes the local copy of the data file (F) upon receiving information about successful proof-of-ownership.

8. The method according to one of the claims 1-7, characterized in that for downloading a data file (F) from the storage entity (S), the client (C) submits the FID to the proxy entity (P), and the proxy entity (P) provides after successful check that the client information of the client (C) matches to the meta-data associated the data file (F) with said FID, server download information to the client (C), preferably wherein the server download information are only temporary valid.

9. The method according to claim 8, characterized in that for decrypting the downloaded data file (F) with a decryption key the client (C) either uses a corresponding cached decryption key associated with the FID or the client (C) performs step a) to acquire the corresponding decryption key.

10. The method according one of the claims 1-9, characterized in that in case the PFID does not match the FID the data file (F) corresponding to the PFID is deleted from the storage entity (S).

11. The method according to one of the claims 1-10, characterized in that the directory operations on a file system of the client (C) are performed locally on the client (C) hidden from the proxy entity (P).

12. The method according to one of the claims 1-11, characterized in that a data file (F) is stored on the storage entity (S) under a random identifier mapped to the FID.

13. The method according to claim 12, characterized in that when a data file is indicated by a client (C) to be deleted the proxy entity (P) renames the data file to another random identifier and provides upon a request for access to the renamed data file by another client (C) a corresponding new access information associated to the FID and the renamed data file.

14. A system for storing data of a client (C) on a storage entity (S), comprising a proxy entity (P),
characterized in that
said proxy entity is adapted to generate wherein the master encryption key is a deterministic function of the data file (F) to be stored based on a hash value (FH) of a hash-function (H) performed on the data file (F) to be stored by said client (C),
said client is adapted to encrypt said data file (F) to be stored using the provided master encryption key, and to compute a hash-tree for the encrypted file (EF) and

said proxy entity (P) is adapted to receive the top-hash of the computed hash-tree as file identification – FID - for the encrypted file (EF), to check whether the FID is already known to the storage entity (S) or not,
in case the FID is not known, the client (C) is adapted to upload the encrypted file (EF) to the storage entity (S) and to the proxy entity (P), and wherein
the proxy entity (P) is adapted to compute a top-hash of the encrypted file – PFID – in case the FID is not known or uses a prior computed FID in case the FID is known and performs a proof-of-ownership-procedure for the encrypted data file (EF) to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file (EF) has been proven, the FID being equal with the PFID is stored on the client (C) together with said hash value (FH).

15. A proxy entity (P) connectable to a storage entity (S) and a client (C), preferably for performing with a method according to one of the claims 1-13 and/or a system according to claim 14, characterized in that
said proxy entity (P) adapted
to generate a master encryption key wherein the master encryption key is a deterministic function of the data file (F) to be stored based on a hash value of a hash-function (H) performed on the data file (F) to be stored by said client (C),
to receive a top-hash of a computed hash-tree as file identification – FID - for the encrypted file (EF),
to check whether the FID is already known to the storage entity or not and
in case the FID is not known, to receive the encrypted file from the client (C),
to compute a top-hash of the encrypted file – PFID – in case the FID is not known or to use a prior computed FID in case the FID is known and
to perform a proof-of-ownership-procedure for the data file (EF) to be stored by comparing the FID with the PFID or the prior computed FID and when the ownership of the data file has been proven,
to indicate to the client (C) that the FID being equal with the PFID.

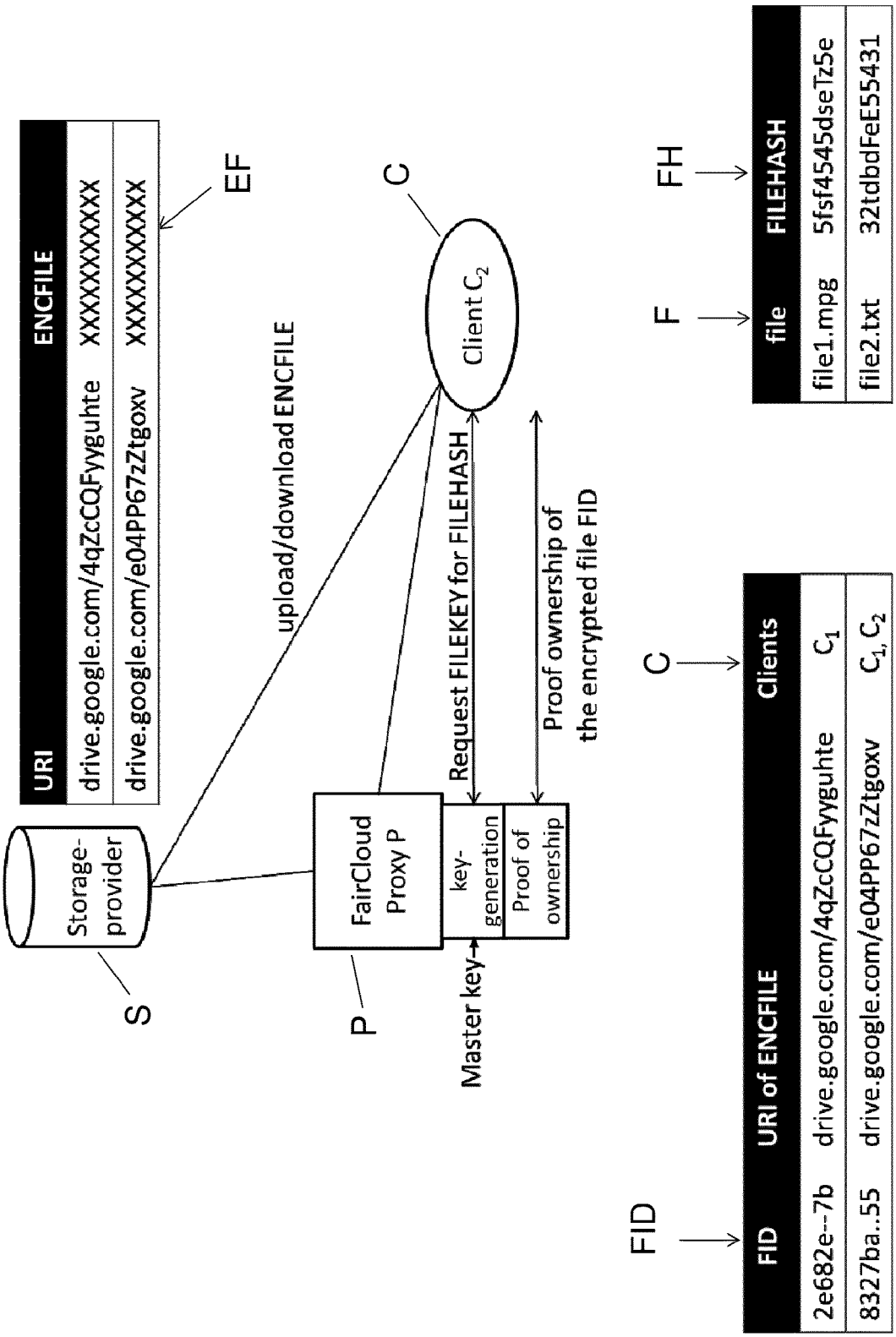

EF
URI
ENCFILE
drive.google.com/4qZcCQFyyguhte XXXXXXXXXX
drive.google.com/e04PP67zZtgoxv XXXXXXXXXX
C
Client C_2
upload/download ENCFILE
Storage-provider
S
FairCloud Proxy P
P
key-generation
Proof of ownership
Master key
Request FILEKEY for FILEHASH
Proof ownership of the encrypted file FID
F
FH
file
FILEHASH
file1.mpg 5fsf4545dseTz5e
file2.txt 32tdbdFeE55431
C
FID
FID
URI of ENCFILE
Clients
2e682e--7b drive.google.com/4qZcCQFyyguhte C_1
8327ba..55 drive.google.com/e04PP67zZtgoxv C_1, C_2

Fig.

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2015/053130

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/62
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JIA XU ET AL: "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage", PROCEEDINGS OF THE 8TH ACM SIGSAC SYMPOSIUM ON INFORMATION, COMPUTER AND COMMUNICATIONS SECURITY, ASIA CCS '13, 1 January 2013 (2013-01-01), page 195, XP055221792, New York, New York, USA DOI: 10.1145/2484313.2484340 ISBN: 978-1-4503-1767-2 last paragraph; page 195, column 2 Section 1.1.4; page 196, column 2 - page 197, column 1 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 October 2015	23/10/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Vinck, Bart

1

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2015/053130

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	MIHIR BELLARE SRIRAM KEELVEEDHI UNIVERSITY OF CALIFORNIA ET AL: "DupLESS: Server-Aided Encryption for Deduplicated Storage", USENIX,, 14 August 2013 (2013-08-14), pages 1-16, XP061014446, [retrieved on 2013-08-15] Section 2; page 180, column 2 last paragraph; page 181, column 2 page 179, column 2, paragraph 2 page 184, column 1, paragraph 3	1-15
A	US 8 528 085 B1 (JUELS ARI [US]) 3 September 2013 (2013-09-03) the whole document	1-15
A	US 8 281 143 B1 (CLIFFORD THOMAS G [US] ET AL) 2 October 2012 (2012-10-02) the whole document	1-15

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2015/053130

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8528085 B1	03-09-2013	NONE	
US 8281143 B1	02-10-2012	NONE	