International Journal of Communication Networks and Information Security (IJCNIS) Vol. 13, No. 2, August 2021

A New Data Deletion Scheme for a Blockchain-based De-duplication System in the Cloud

Yassine El Khanboubi[1], Mostafa Hanoune[2], Mohamed El Ghazouani[3]

[1,2]Faculty of Science Ben M'Sik, Hassan II University,Casablanca, Morocco
[3]Faculty of Science Semlalia, Cadi Ayyad University, Marrakesh, Morocco

Abstract: Almost all Cloud Service Providers (CSP) takes a principled approach to the storage and deletion of Customer Data. Most of them have engineered their cloud platform to achieve a high degree of speed, availability, durability, and consistency. Their systems are designed to be optimized for these performance attributes and must be carefully balanced with the necessity to achieve accurate and timely data deletion. Many researchers have turn their focus toward data storage and how it will be a challenging task for CSPs in term of storage capacity, data management and security, a considerable number of papers has been published containing new models and technique that will allow data De-duplication in a shared environment but few of them have discussed data deletion. In this paper we will be discussing a new approach that will allow a smart deletion of data stored in the file system as well as its reference in the Blockchain since, by its nature, Blockchains does not allow deletion without violating the Blockchain's consistency, a preexisting de-duplication system will be our base platform on which we will be working to achieve an accurate and secure data deletion using Blockchain technology while preserving its consistency**.**

Keywords— Blockchain, Data Deletion, De-duplication, Cloud Computing

1. Introduction

Data de-duplication is a set of techniques implemented in order to reduce the amount of data in term of capacity in storage systems by detecting and eliminating redundant data and has been widely used for data backup to minimize bandwidth and storage overhead. Depending on the implemented solution, the system will perform a set of actions which mainly verifies whether the file in question (or part of the file called file block of file chunk) is already stored in the file system or not.

There have been a number of de-duplication systems proposed with various de-duplication strategies such as file-level or block-level de-duplications depending on the size of de-duplication. Many innovative approaches has been proposed such as the implementation, Multi-Agent Systems [24][25] and Blockchain technology with the implementation of different chunking techniques such as content aware chunking algorithms, de-duplication can also be organized in form of 2 types, inline de-duplication system and post-process de-duplication system, which performs de-duplication before or after storing them respectively. Another form of categorization is the location where de-duplication is verified, the Client side or the server side.

With the advent of cloud storage, data de-duplication techniques attract more and more attention from both academic and industrial community [26]. It has become one of the main techniques for the management and the reduction of the ever-increasing volume of data in the information society. According to the analysis report of IDC, IDC predicts that by the year 2025 the Global Datasphere will grow from 33 ZB (Zettabytes) in 2018 to 175 ZB and that public cloud environments will hold 49% of the world's stored data. The cloud storage market size was valued at $46.12 billion in 2019, and is projected to reach $222.25 billion by 2027, growing at a CAGR of 21.9% from 2020 to 2027. Today's commercial cloud storage services, such as Dropbox, Mozy, and Memopal, have been applying and currently racing toward improving their de-duplication system.

The data storage supply and demand worldwide from 2009 to 2020 is presented in Figure.1 below:

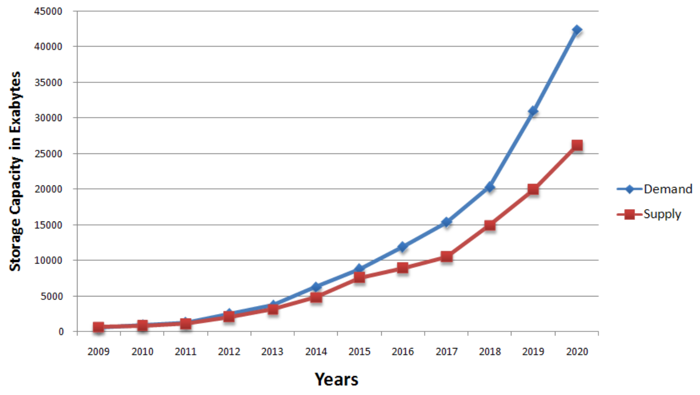


Figure 1. Data storage worldwide supply and demand from 2009 to 2020

De-duplication being the most effective approach for saving storage space can be improved by other techniques such as data deletion which is a key aspect in maintaining the occupied storage space as low as it can be which will benefit the cloud services in term of speed, availability, durability, and consistency.

Many papers had discussed data de-duplication and deletion in terms of security, bandwidth and storage space, and proposed some efficient models that (in numbers) helped reducing considerably the amount of bandwidth and storage space needed, but left many issues related to the proposed model undiscussed.

In [1] El Khanboubi and Hanoune proposed a new framework based on Blockchain technology where de-duplication is made on the client side and the original file is chunked using a content-aware chunking algorithm in order to increase the probability to de-duplicate not only files in whole but even parts of files, the actual file (or chunks of the file) is stored in cloud storage while its references are stored in a Blockchain, the proposed structure of the Blockchain allow user having uploaded the same file to share only references without uploading the same file again, and if two files share one or more part of a file, only the missing parts are uploaded. This model improves significantly the amount of data stored, but knowing that Blockchain technology is immutable and doesn't allow deletion, this approach does not allow blocks deletion thus file deletion will leave an incoherent system, a corrupted data even.

El Ghazouani et al [2], proposed a similar approach for ensuring data integrity auditing where a network of thin clients that run from resources stored on a central server instead of a localized hard drive, this server also plays an important role as the network mediator, which means no data needs to be transferred from clients to the mediator, de-duplication according to their approach is performed on the client side, only the missing files or file blocks are uploaded to the cloud, on the other hand, the CSP must –after a successful upload – update the Blockchain by adding a new block containing the information about the new uploaded file (or file blocks). With the references being stored in the Blockchain, the same deletion issue as the previous model will emerge.

In this paper we will discuss a new deletion scheme for a Blockchain-based de-duplication system that will improve upon the previously described framework while preserving data integrity

2. Related Works

In their work, Martin Florian et al. presented a specific proposal towards Functionality-Preserving Local Erasure (FPLE) [3]. The FPLE as described is implemented as an extension to existing node software for common Blockchain networks like Bitcoin. FPLE enables individual node operators to mark chunks of data (e. g., transaction outputs) for erasure without requiring any update or modification on the protocol, coordination with other nodes or the introduction of global trust anchors.

In their proposed model, they have implemented a separate database called ❛❛the erasure database❜❜ which is a key-value store mapping Transaction IDs (TXID) to redacted transactions, i.e., for the transaction T with TXIDi_T it stores the tuple (i_T, T ❛). Additional data, such as the transaction's block's hash, can be included as well in the implementation.

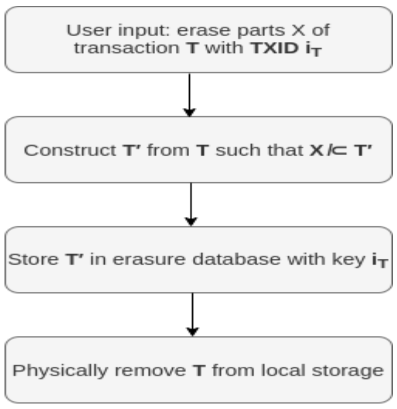

User input: erase parts X of transaction **T** with **TXID i_T**
Construct **T'** from **T** such that **X $\not\subset$ T'**
Store **T'** in erasure database with key **i_T**
Physically remove **T** from local storage

Figure 2. Erasure of transaction parts

2.1 Avoiding Unwanted Data

In [4], Matzutt et al. discuss numerous approaches for preventing the insertion of random, potentially unwanted data onto cryptocurrency Blockchains. Their proposed work includes content detectors, which have the ability to filter transactions based on heuristics and knowledge about frequently used data insertion methods, as well as protocol adjustment that would greatly increase the costs of including random data. Various approaches along these lines have also emerged in the non-academic cryptocurrency community. Techniques aiming to avoid the insertion of unwanted data depend on global adoption, for an effective filtering, and in some cases also on protocol changes when applied to preexisting networks. In contrast, FPLE requires only a nodelocal decision and is in this way both more practical and enables the incorporation of a wider range of individual preferences and constraints. Lastly, as it can be seen in similar application domains such as digital rights protection or malware detection via upload filtering, content-based filtering is never fully circumvention-proof. Once something "slips through", an erasure possibility again becomes mandatory when taking data protection into consideration as a reason for erasure, it is also noteworthy that a respectable number of researchers focused their interest toward the challenge of providing anonymity to Blockchain users (see e. g. [5] for a recent survey). However, most transactions in popular systems like Bitcoin do not use any additional techniques to increase anonymity [6] and are re-identifiable using various well-known techniques [5]. Even when strong privacy can provide a certain degree of guarantees that can be achieved through technical means, this provides no solution for cases where identifiable data is published to the Blockchain on purpose, e. g., as part of "doxing".

2.2 Redacting the Blockchain

A simple, yet effective, approach to globally delete previously inserted data from a Blockchain is to produce a hard fork [7]. Safe hard forks require a strong off-chain consensus among all miners, users and network operators. In public networks with little central coordination, such as Bitcoin, such a consensus is notably difficult to achieve. Even more so when compared to the ease of including potentially problematic data at a high rate. Redactable Blockchains [7] have been proposed as an alternative Blockchain design that allows the global erasure of data without causing hard forks. They use "chameleon" hash functions [8] that enable trusted entities with access to a trapdoor key to calculate hash collisions and therefore change posted data while maintaining the appearance of chain integrity. Numerous alternative solutions were proposed that deal with resulting trust problems by using a voting-like approach [9], [10]. However, [7], [9] and [10] require heavy changes to existing systems, also altering their underlying trust model. In contrast to their motivation of erasing data globally, in our work, we focus on local erasure without requiring protocol changes.

2.3 Pruning

Pruning is a commonly used technique for erasing older parts of a Blockchain, mainly with the aim of reducing cloud storage requirements. While related, our erasure technique differs in its objective - we erase data possession relationship before erasing the whole history at a certain stage and only deleting non-shared file chunks - and provide solutions for distinguished challenges such as the pruning of data potentially relevant for validating future blocks. The latter challenge is highly relevant in practice as problematic data is often encoded in unspent but potentially spendable transaction outputs [11]. Different Blockchain designs such as [12] propose storing the current global state, e. g., in terms of account balances, in each block so that older blocks can be more safely pruned.

A per-block cryptographic engagement to the current state is also used in popular Blockchain based networks such as Ethereum [13]. While potentially making older transactions more easily prunable, neither of these solutions help in cases where potentially unwanted data can be rebuilt from the current state, such as when it is encoded in account addresses or SmartContract data. With FPLE, they explicitly consider the erasure of data that is part of the UTX.

3. Preliminaries

In this paper, we have adopted several concepts and models proposed in various papers and researches, and also technologies that were implemented to ensure our proposed model efficiency, so before getting into discussion about our proposed model, below are the main concepts used in it.

3.1 Cloud service models

There are usually three models of cloud service to compare: Software as a Service (*SaaS*), Platform as a Service (*PaaS*), and Infrastructure as a Service (*IaaS*). Each model of these three has its benefits according to your needs, as well as variances, before choosing a model, it is necessary to understand the differences between *PaaS*, *Saas* and *Iaas*:

• **PaaS** (Platform as a Service) delivers a computing platform where the client can create, execute, deploy and manage their applications.

• **SaaS** (Software as a Service) is a model of software deployment where the software/applications are provided to the customers as a service through a program interface or a web browser. The Cloud's client does not need to install IT infrastructure such as network, servers, operating systems and application software inside his company because they are hosted and managed in supplier's site.

• **IaaS** (Infrastructure as a Service) delivers computer infrastructure, basically a virtualization environment platform as a service. Rather than purchasing servers, software, storage, memory, processor or network equipment, clients buy those resources as fully outsourced services.

3.2 Blockchain

3.2.1 Definition

Since the introduction of Blockchain technology by Satoshi Nakamoto in 2008 [14], its popularity grew thanks to its efficiency in ensuring data integrity and security, Blockchain is mostly a public ledger or a distributed database that holds every event or transaction that is distributed to all participants, Blockchain can store any type of data, Generally, Blockchain can be divided into two groups – public Blockchain and private Blockchain. As a novel data structure, both public and private Blockchain have two attractive advantages. By generating a token as a chain of transactions, Blockchain makes it possible to deal with the problem of double spending in a distributed network.

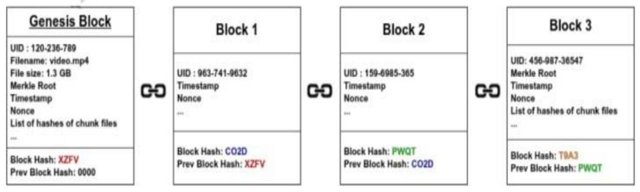


Figure 3. Structure of a Blockchain used in our proposal

Each block in the Blockchain must have a reference to its precedent; this reference is the hash of the previous block.

The first block in a Blockchain is called Genesis Block (block #0) which has no previous block hash, this block can be very useful in our case, it will be used as a reference of all transaction, new owners, updates (which will be discussed in a future paper) … of a unique file on the server's file system.

Blockchain could be considered as a form of distributed databases. However, it is different from existing regular distributed databases (such as HBase, MondoDB…) by two main features:

1- Cryptography

Authenticity, user identity and the ledger integrity are achieved using encryption [23].

2- Decentralization

Unlike regular databases and ledgers, the Blockchain security and functionalities enforced in a public and distributed way instead of relying on a centralized server or a central authority.

First, what exactly are the possibilities at hand? To summarize, there are generally three categories of Blockchain-like database applications.

<u>*Public Blockchains*</u>: a public Blockchain is a Blockchain that anyone can read, anyone in the world can submit transactions to and expect to see them added if they are valid, and any individual can participate in the consensus process - the process for determining what blocks get added to the chain and what the current state is -. As a substitute for centralized or quasi-centralized trust, public Blockchains are secured by cryptoeconomics - the combination of economic incentives and cryptographic verification using mechanisms such as proof of work or proof of stake, following a general principle that the degree to which someone can have an influence in the consensus process is proportional to the quantity of economic resources that they can bring to bear. These Blockchains are generally considered to be "fully decentralized".

<u>*Consortium Blockchains*</u>: a consortium Blockchain is a Blockchain where the consensus process is managed by a pre-selected set of nodes; for example, one might consider a consortium of 20 banks, each of which operates a node and of which 15 must sign every single block in order to validate the block. The right to read the Blockchain may be restricted or public to the participants, and there are also hybrid routes such as the root hashes of the blocks being public, together with an API that allows public members to make a limited number of queries and receive cryptographic proofs of some parts of the

Blockchain state. These Blockchains may be considered "partially decentralized".

Private Blockchains: a fully private Blockchain is a Blockchain where write permissions are kept centralized to one organization. Read permissions may be public or restricted to a random extent. Similar applications include database management, auditing, etc internal to a single company, and so public readability may not be necessary in several cases at all, though in other cases public auditability is preferred.

3.2.2 Proof of Work

The proof of Work concept has emerged in 1993, when Cynthia Dword and Moni Naor published a paper [15] where they introduced a new method that tries to prevent spam emails.

A proof of work is a consensus algorithm in which it is costly and time-consuming to produce a piece of data, but it is easy for others to verify that the data is correct. Bitcoin is using a proof of work system called "HashCash" [16].

For a block to be accepted by the network, miners have to complete a proof of work that comes in the form of an answer to a mathematical problem (challenges) to verify all transactions in the block which in order to solve this problem, nodes must run a long and random process. The proof of work difficulty of is not always the same and may vary, it keeps adjusting in order to generate new blocks every 10 minutes. The probability of a successful generation is very low, what makes it unpredictable which worker in the network will generate the next block.

3.3 Content-aware Chunking

chunking is a crucial step in order to achieve data de-duplication and also to lighten the outgoing network traffic during file upload, as described in various papers, chunking is one of the main challenges in the de-duplication system and can be achieved using different methods [17]:

- Content-aware chunking.
- Fixed-size chunking.
- File-level chunking.
- Robin Chunking.
- Two Threshold Two Denominators (TTTD).
- Bimodal chunking.
- …

In the proposed model, a file chunk can be part of one or many files which will reduce the storage space needed; in this case a simple binary chunking (also known as Fixed-Size chunking that splits a file into equally sized chunks) will reduce the probability of having two files that share the same set of data. The chunks will be just a sequence of bits and boundaries are based on offsets like 4, 8 or 16 KB, which reduce the probability of using the same chunk in other files. Instead, using a Content-aware chunking algorithm will split the file based on its content, which will considerably improve the chunk re-usage probability.

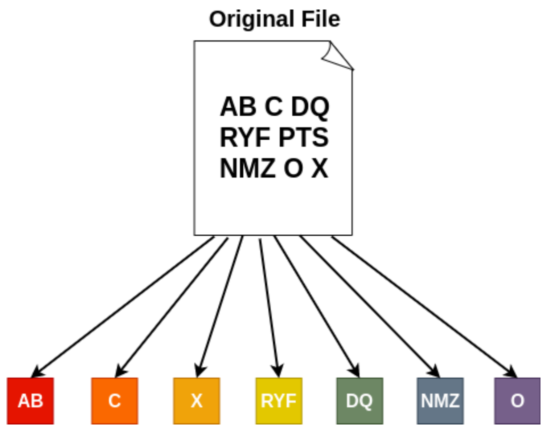


Figure 4. Content-Aware chunking based on "Text" type

3.4 Merkle Tree

In cryptography, a Merkle Tree - also called Binary Hash Tree [18] is a type of data structure that is used to verify the integrity of a large amount of data. Technically speaking, the Merkle tree is a binary tree where leaves are data blocks hashes, and every non-leaf node is the hash of the concatenation of two child blocks (using more than 2 child nodes is also possible but the majority of Merkle tree implementations are binary), the process is repeated several times according to the number of starting leaves until it gets a unique hash called Merkle Root. Consider a file split into 4 parts P1, P2, P3 and P4.The hashes are computed as follow: H1 = h(P1), H2 = h(P2), H3 = h(P3), H4= h(P4), H5 = h(H1 + H2), H6 = h(H3 + H4), the Merkle root: H7 = h(H5 + H6),where h() is a double-hash function defined as h(x) = MD5(MD5(x)) and "+" means the concatenation hashes strings, the process will generate a hash tree of 3 levels.

The main advantage of the Merkle tree is that the slightest modification in one or more nodes will result in a whole new tree and the Merkle root will change completely, which will be very advantageous in verifying data integrity. A Merkle tree needs an even number of nodes in every level according to its depth in order to generate the hashes of a higher level in the tree, if an odd number of nodes is found while generating the tree, the last node is duplicated.

The tree depth is not (and can never be found) indicated by the Merkle Root, this way a "second-preimage attack" [19] (where an attacker can create a file different than the original one that has the same Merkle hash root) can never be executed.

In our case, a file will be chunked into small parts and then the Merkle tree will be generated based on the resulted chunks (the number of levels in the Merkle tree depends on the initial number of chunks generated by the Content-Aware chunking algorithm) as shown in the figure below.

Any Hashing algorithm is applicable, we have decided to use MD5 algorithm.

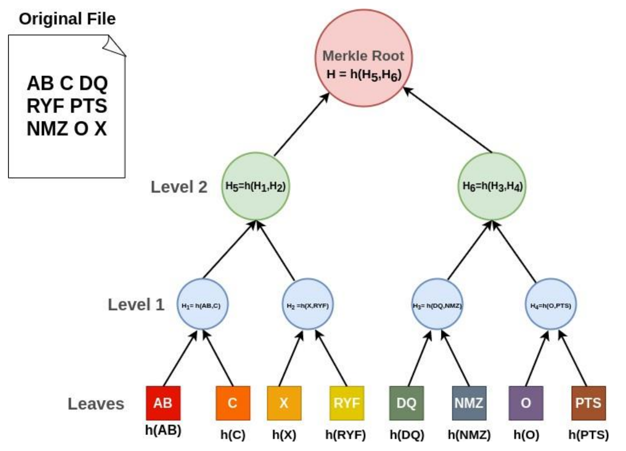


Figure 5. Merkle tree generated based on the file chunks.

4. Description of the Proposed Model

In this section we will describe our proposed model which is based on a preexisting model that uses Blockchain technology to improve Data upload and storage space in Cloud Computing [1].

4.1 GDPR and the "Right to be forgotten"

General Data Protection Regulation (GDPR) [20] is a regulation in European Union law on data privacy and protection in the EU and the EEA (European Economic Area). It has become enforceable since 25 May 2018and also addresses the transfer of personal data outside the EU and EEA areas. A key aspect of the GDPR on Blockchain is the fact that personal data is not to leave the European Union. This is a serious issue with public Blockchains, since there is no control on what entity hosts a node. This is less an issue when it comes to permissioned or private Blockchains. To tackle this problem, The Interplanetary Database (IPDB) set up a foundation that could guarantee data stays in the EU because it is publicly accessible (client side) but permissioned hosted (node or cloud side) Blockchain .

The GDPR's main goal is to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the European Union. It has a major impact in organizations both large and small.

Article 17 of the GDPR explains how individuals have the right to have personal data permanently erased. This is also known as the 'right to be forgotten' [21]. The right is not absolute and only applies in certain known circumstances.

Due to GDPR, storing personal data on a Blockchain is not an option anymore. A popular option to get around this problem is a very simple one, you store the personal data off-chain and store the reference to this data, along with its hash and other metadata (like ownership status, timestamp, permissions…), on the Blockchain. This approach was implemented by El Khanboubi and Hanoune in [1] which makes the proposed model partially GDPR compliant since the data is not fully erased when requested and the ownership history can still be retrieved if there are other users owning the same file.

To make this approach fully GDPR compliant, we will be discussing a new layer that will be added to the upload process aforementioned in the this section.

4.2 The proposed model

The model in question [1] uses several techniques and concepts that were aforementioned. The authors proposed the usage of a Blockchain for each file newly uploaded; before storing it in the file system, the file in question goes through a two-steps process as follow:

Step 1 (the Client side): before uploading it to the cloud, the file is broken up using a Content-Aware chunking algorithm into multiple pieces called chunks, this type of algorithm helps increasing the probability of using the same chunk in other files and thus reducing the storage space needed. The client then generates the Merkle Tree which is then sent to the Cloud.

Step 2 (the Server Side): once received, the Merkle tree is then looked up in all Blockchains, if it is found the servers then checks the ownership, if the user already owns the file then nothing is to be done and the client is informed if not, the an ownership block is added to the Blockchain, another case is if the Merkle tree is not found, in this particular case the servers asks the client for the list of hashes generated based on the file chunks, once received, the servers check for duplication and generates a list that contains only the missing parts to be uploaded. The client then starts the upload process and the server creates a new Blockchain.

With the implementation of this approach, two major issues emerged. Knowing that the Blockchain is immutable and therefore cannot be modified or altered, file updates and deletion is impossible, our work in this paper will focus on data deletion using the previously described model.

Since the Blockchain cannot be modified, we decided to update structure of the added blocks in order to add a "state" field, this field will define the current stat of the file in question and its value should be one of two stats:

• ***Original***: this state means that the file for the current user is still valid and available for the user to access.

• ***Updated***: A block in the "Updated" state means that a user owning the file has submitted a new update to the file, in this paper we're only highlighting the update state for it is left for a future work and won't be discussed.

• ***Deleted***: this state means that the file has been deleted by the user; a file marked as deleted by a given user will be unavailable and can no longer be accessed, other users owning the same file will still be able to access it. If the file is owned only by the user from where the deletion request originated this field will not be used upon deletion.

Since the genesis block (the first block of the Blockchain) is generated by the first user who uploaded the file its state is always set to Original.

The new structure of the blocks constituting the Blockchain is described in Figure 6 below:

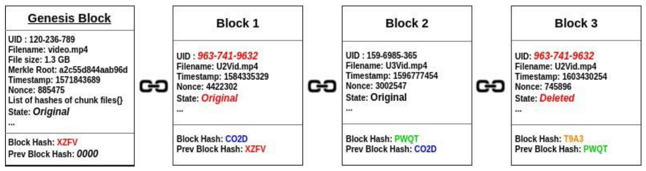


Figure 6. Structure of a Blockchain used in our proposal

While every cloud user has a set of blocks mined in different Blockchains for different files, file deletion technique will be changed according to different use cases depending on the user and ownership of files.

Figure 6 represents the new edited structure of the Blockchain proposed in our work, notice that the “state” field has been added to each block, and also Block 1 and Block 3 are mined by the same user, in block 3 the file state has been changed to “Deleted” which means that the user (with the User ID 963-741-9632) has deleted his ownership of the file.

4.3 Securing the process

In order to make this model more secure and fully GDPR compliant two more steps has been added to the upload and deletion process.

- Block Encryption

Since every block in the Blockchain holds information about users ownership and deletion of a file, which is -in certain cases- only made by altering the state of a file and therefore skimming the Blockchain can give you the full history of every user ever owned a file. To remedy this issue we have decided to use asymmetric encryption to encrypt the metadata that will be stored in the new generated block. As described in [1], the CSP already holds the user’s public key that is used during the authentication process, this way the user’s identity as well as his files history of ownership will be preserved.

To retrieve the data, the user must first authenticate using his private key, the user should hold (in a local database on the client side) the number of every block that was mined in the Blockchain on his behalf by the CSP (knowing that users are not allowed to add new block to the Blockchain), this way when accessing a file, the user does not have to iterate over each block trying to decrypt one by one but instead he decrypts only the data that was encrypted using his private key.

The “state” field should not be encrypted, as described earlier, the “state” field is mandatory to determine whether to delete the whole Blockchain or generate a new block to add to it, all users should be able to check if the file is still owned by other users.

The Encryption/Decryption process is illustrated in figure 7.

- Proof of Ownership (POW)

Knowing that a file could be owned by multiple users, file deletion should be an accurate and secure process that prevents data loss and revoke access to malicious clients.

Despite the strength provided to our model by the key aspects that we have already implemented (block encryption, Merkle hash tree, Content Addressing), we have decided to add an extra security layer to our system called Proof of Ownership (POW).

POW schemes are security protocols designed specifically to allow a server to verify (with a certain degree of assurance) whether a user owns a file. The probability that a malicious client engages in a successful POW run must be negligible in the security parameter, even if the malicious client knows a (relevant) portion of the target file. A POW protocol should be efficient in terms of CPU usage, bandwidth and I/O for both the server and all legitimate clients, in particular, POW schemes should not require the server to load the entire file (or a large portions of it) from its back-end storage at each execution of POW. Additional assumptions about POW schemes are that they should take in consideration the fact that a user wishing to engage in a successful POW run with the server may be colluding with other users who own the same file and are willing to help in circumventing POW checks. These latter users, however, are neither assumed to always be online (i.e. they cannot answer the POW challenges on behalf of the malicious user), nor are they willing to exchange very large amounts of data with the malicious user. Both assumptions are arguably reasonable, as such users would have no strong incentive in helping the free-riders

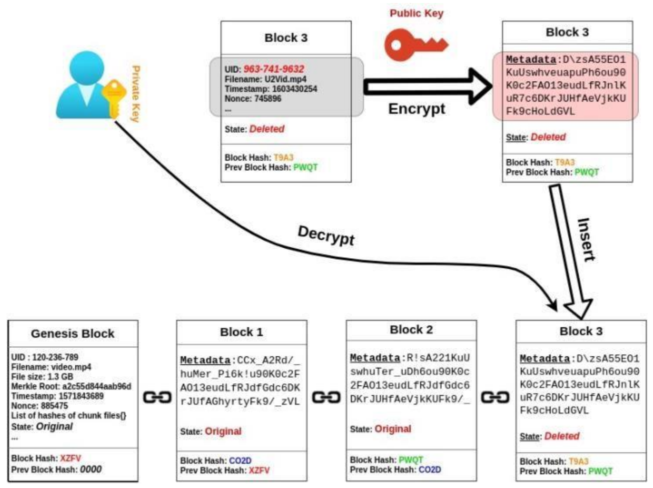


Figure 7. Encrypting/Decrypting blocks Data

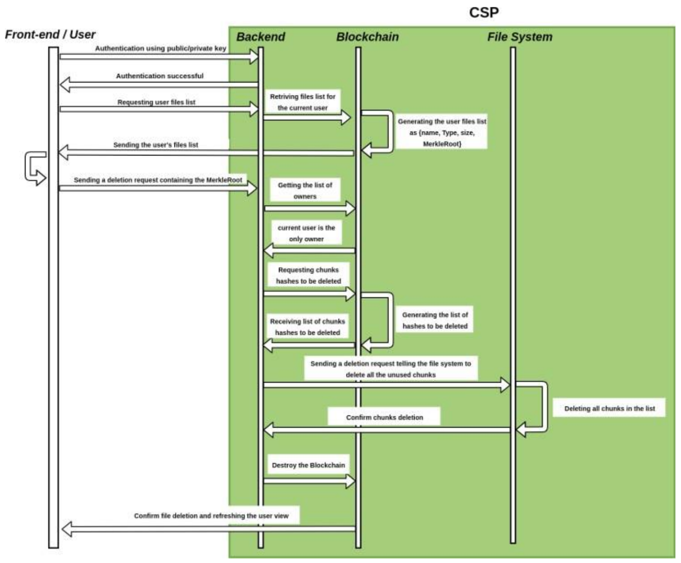
4.4 Analysis and Discussion

Following the logical course of actions, different use cases emerge and are described below:

Case 1: if a user has just uploaded a file that was not previously uploaded by another user and then changed his mind and decided to delete it.

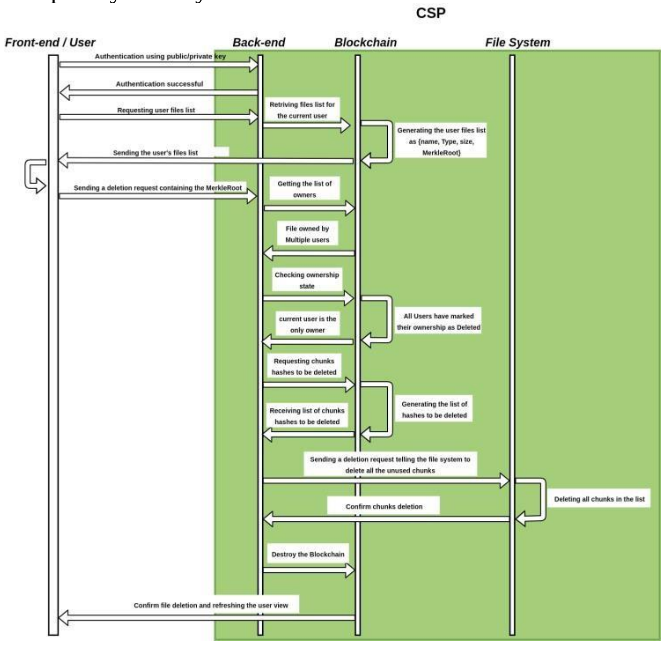
In this case the user uploaded a file that doesn’t exist on the Cloud storage (as described in [1], the file is looked up using the Merkle tree that was generated on the client side and then sent to the server before uploading it), once the upload process is completed the user decided to delete the file.

By uploading the file, the back-end application created the Genesis Block of a new Blockchain dedicated to the newly uploaded file and assigned the user that lunched the upload process as the first owner of the file, this Blockchain will hold all the information that will be used to manage ownership and data de-duplication, in this case, deleting the files means also deleting the newly generated Blockchain, upon the reception of the deletion request, the CSP will check the identity of the user and his ownership of the of the file, once confirmed the CSP will then get the list of the hashes of all chunks constituting the given file, the server will iterate over the list and check if each chunk is part of another file or not, all chunks that are unique to the file in question will be deleted from the cloud storage, the Blockchain will then be completely destroyed.


CSP
Front-end / User
Backend
Blockchain
File System
Authentication using public/private key
Authentication successful
Requesting user files list
Retriving files list for the current user
Generating the user files list as {name, Type, size, MerkleRoot}
Sending the user's files list
Sending a deletion request containing the MerkleRoot
Getting the list of owners
current user is the only owner
Requesting chunks hashes to be deleted
Generating the list of hashes to be deleted
Receiving list of chunks hashes to be deleted
Sending a deletion request telling the file system to delete all the unused chunks
Deleting all chunks in the list
Confirm chunks deletion
Destroy the Blockchain
Confirm file deletion and refreshing the user view

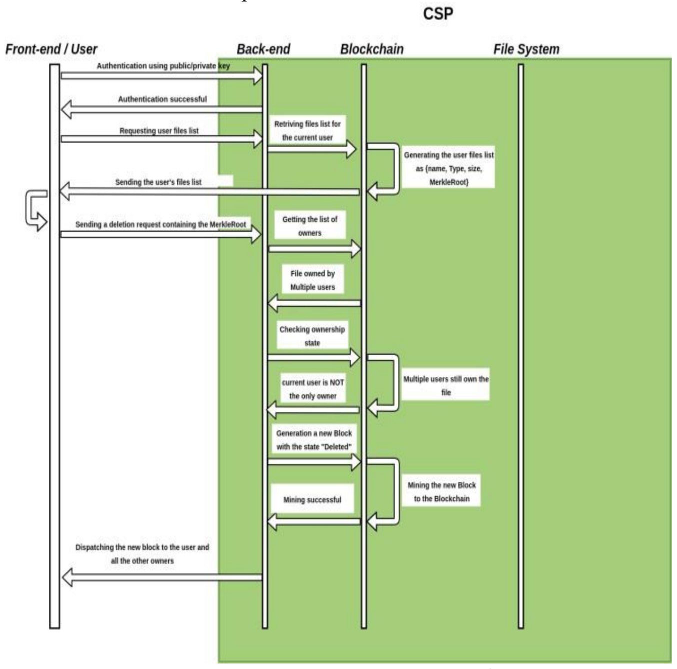
Case 2: if the user owns alongside with other users the same file and all the other users have already deleted their file.

In this case the user owns a file alongside with a given number of other users, all the other users have deleted their ownership of the file (every user owning the file have submitted a deletion request and mined a new block to the Blockchain with the state "Deleted") except the current user, this case is similar to the previous one with a slight change in the verification process, the CSP must first run through every block in the Blockchain to verify the there is no other user owning the file, after the verification the CSP will then get the list of the hashes of all chunks constituting the given file, the server will iterate over the list and check if each chunk in the list is part of another file or not, all chunks that are unique to the file in question will be deleted from the cloud storage, the Blockchain will then be completely destroyed.


CSP
Front-end / User
Back-end
Blockchain
File System
Authentication using public/private key
Authentication successful
Requesting user files list
Retriving files list for the current user
Generating the user files list as {name, Type, size, MerkleRoot}
Sending the user's files list
Sending a deletion request containing the MerkleRoot
Getting the list of owners
File owned by Multiple users
Checking ownership state
All Users have marked their ownership as Deleted
current user is the only owner
Requesting chunks hashes to be deleted
Generating the list of hashes to be deleted
Receiving list of chunks hashes to be deleted
Sending a deletion request telling the file system to delete all the unused chunks
Deleting all chunks in the list
Confirm chunks deletion
Destroy the Blockchain
Confirm file deletion and refreshing the user view

Case 3: if the file remains owned by several users at the same time including the given user.

In this case the user owns a file alongside with a given number of other users, few or none of the users have already sent a deletion request, in this case a new block with the state "***Deleted***" will be added to the Blockchain for the current user, no further action is required.


CSP
Front-end / User
Back-end
Blockchain
File System
Authentication using public/private key
Authentication successful
Requesting user files list
Retriving files list for the current user
Generating the user files list as {name, Type, size, MerkleRoot}
Sending the user's files list
Sending a deletion request containing the MerkleRoot
Getting the list of owners
File owned by Multiple users
Checking ownership state
Multiple users still own the file
current user is NOT the only owner
Generation a new Block with the state "Deleted"
Mining the new Block to the Blockchain
Mining successful
Dispatching the new block to the user and all the other owners

With this approach, the owner can delete his file at any time by setting the "*State*" field to "***Deleted***" without violating the Blockchain's consistency.

5. Implementation and Algorithms

As mentioned earlier in this paper, our work aimed to improve upon the model proposed in [1] by El Khanboubi and Hanoune in order to remedy some major issues regarding security and file deletion that wasn't discussed in the previous work, the model was implemented using various technologies for the front-end application as well as the back-end application.
Based on the aforementioned implementation we will be adding a minor upgrade to the existing code as well as adding an entire new layer to support the file deletion process, our new implementation is described as follow:

- An upgrade to the Blockchain structure to support "States"
- An upgrade to the authentication system to support key-based authentication
- An upgrade to the front-end application to support file deletion
- A new Back-end layer to manage file deletion including data encryption, block mining and file deletion from the file system.

Before diving into the details of the upgrade implemented to support the deletion process, we should note that the upcoming algorithms were considerably simplified for the sake of this paper and will be categorized into two major categories, the front-end algorithms and the Back-end algorithms.

5.1 The proposed Block structure

The new edited Blockchain format has been described in section 5 with the addition of a new field called “State” that will define the current state of a file; the figure bellow will describe with more details the new format after applying encryption to preserve the user’s identity

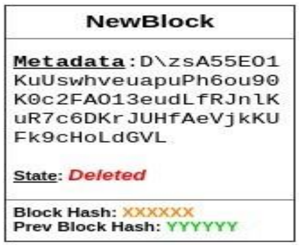


Figure 8. The proposed block format

The “*Metadata*” field holds all the data about the user and his ownership in an encrypted format that only he can decrypt using his private key.

5.2 The Front-end Algorithms

The front-end application performs different tasks according to which phase we’re in, the first thing to do – after the authentication process that we will not be describing as an algorithm – is to update the user interface to show the list of all the files that the user owns, since the user has a copy of each Blockchain of each file he owns, there is no need to request the cloud server to fetch the files list, algorithm 1 bellow shows the steps performed to refresh the UI for the user to interact with.

Algorithm 1. User’s files listing

```
1 for blockchain in localBlockchainList
1     filesHashesList[] = Blockchain
                          → getGenesisBlock()
                          → MerkleTree
2 endFor
3 FilesList[] = submitHashes(filesHashesList);
4 foreach file in filesList
5     UIBlock = new UIBlock();
6 UIBlock → setName(file → name)
7     UIBlock → setType(file → type)
8 UIBlock → setSize(file → size)
9 UIBlock → setID(file → MerkleTree)
10 UIBlock → setFileIcon()
11 Panel → add(UIBlock)
12 endForeach
```

Once the user interface is available, the user can then interact with it and choose a file to delete, by clicking the ‘Delete’ button a two stages process is then triggered and starts with proving to the server that the current user is indeed a legitimate owner of the file in question by solving a set of challenges defined by the server referred to as “Proof of Ownership” that we will be describing in the next section, on the other hand, the client should fire a set of action after completing the aforementioned challenges, Algorithm 2 describes these actions.

Algorithm 2. Deletion Event

```
1 DelButton → ClickEvent(
1   fileID = this → getID()
2   submitDeletionRequest(MerkleRoot)
        → done(status, callback(
3               if status is 'success' then
                    Panel → removeUIBlock(fileID)
4               else
                    showError()
5               endIf
6           )
7       )
8 )
```

5.3 The Back-end Algorithms

As described in the beginning of this section, the back-end software should be upgraded in order to support the newly implemented functionalities; the first main layer that will strengthen our security protocol is the “Proof of Ownership” process, since our approach already uses the Merkle Tree concept we have adopted two challenges to be satisfied by the client:

- The server regenerate the Merkle Tree of a file and then compare it with the one submitted by the user.
- The user must be able to decrypt at least one block in the Blockchain.

The ‘Proof of Ownership’ algorithm is described below:

Algorithm 3. Proof of Ownership

```
1  CltMerkleRoot = request → MerkleRoot;
1  chunksList[] = getBlockchain(MerkleRoot)
                     → getBlock(1)
                     → chunksHashes();

2  SrvMerkleRoot = reGenMerkleRoot(chunksList);

3  if CltMerkleRoot == SrvMerkleRoot then
4    response → Challenge1Status('success');
5  response → send();
6  else
7  response → Challenge1Status('fail');
8  exit();
9  endIf

10 if request → DecryptOneBlock() == true then
11   delete();
12 response → Challenge2Status('success');
13 else
14   response → Challenge2Status('fail')
15 exit();
16 endIf
```

Last but not least, after successfully proving the user's ownership of the file in question, a deletion process should be triggered depending on the correct use case as described in ***Section 5.2***.

6. Conclusion

Knowing that Blockchain technology is immutable and the data that holds cannot be altered or erased, our approach demonstrated its effectiveness as a workaround to this issue while preserving the user's identity, using encryption there is no way for any third party or even the CSP to identify or track a user's history of owning a file which makes our proposed model GDPR compliant.

Also, as an extra security layer, we have implemented the concept "Proof of Ownership" in a de-duplication system to makes it more efficient, POW challenges is a set of tasks that should be mathematically solved in order to confirm an ownership, those challenges can easily drain the server's resources, depending on the file's type and size but it is a mandatory step toward a privacy preserving system as well as an extra security layer to guarantee data consistency.

As efficient as it seems, this model can only be used as an archiving system since there is no way to update the content of a file stored using this approach. This is due to use of Blockchain technology that is known for its immutability to store the file's metadata and references.

Our future work focuses on the possibility of updating a file's content while preserving the consistency of the chain.

Another major issue that emerged with the approach discussed in this paper is computational power. Encryption, decryption, solving challenges and deletion are all heavy tasks for a regular CPU to perform especially on the client side since the hardware resources are usually limited and easily drained by heavy tasks.

References

[1] Y. El Khanboubi and M. Hanoune, "Exploiting Blockchains to improve Data Upload and Storage in the Cloud," IJCNIS, vol. 11, no. 3, pp. 357–364, 2019.

[2] M. El Ghazouani, M. A. El kiram, E.-R. Latifa, and Y. El Khanboubi, "Efficient Method Based on Blockchain Ensuring Data Integrity Auditing with Deduplication in Cloud," Int. J. Interact. Multimed. Artif. Intell., vol. 6, no. 3, p. 32, 2020, doi: 10.9781/ijimai.2020.08.001.

[3] M. Florian, S. Henningsen, S. Beaucamp, and B. Scheuermann, "Erasing Data from Blockchain Nodes," Proc. - 4th IEEE Eur. Symp. Secur. Priv. Work. EUROS PW 2019, no. April, pp. 367–376, 2019, doi: 10.1109/EuroSPW.2019.00047.

[4] R. Matzutt, M. Henze, J. H. Ziegeldorf, J. Hiller, and K. Wehrle, "Thwarting unwanted blockchain content insertion," Proc. - 2018 IEEE Int. Conf. Cloud Eng. IC2E 2018, pp. 364–370, 2018, doi: 10.1109/IC2E.2018.00070.

[5] M. Conti, K. E. Sandeep, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," IEEE Commun. Surv. Tutorials, vol. 20, no. 4, pp. 3416–3452, 2018, doi: 10.1109/COMST.2018.2842460.

[6] M. Möser and R. Böhme, "Anonymous alone? Measuring Bitcoin's second-generation anonymization techniques," Proc. - 2nd IEEE Eur. Symp. Secur. Priv. Work. EuroS PW 2017, pp. 32–41, 2017, doi: 10.1109/EuroSPW.2017.48.E. Soja, C. Wade, C. Duncan, and J. Clampett, "Rewriting History in Bitcoin and Friends," 2017, [Online]. Available: https://eprint.iacr.org/2016/757.pdf%0Ahttp://www.vba.vic.gov.au/da ta/assets/pdf_file/0005/19553/Fire-protection-for-high-rise-buildings.pdf.

[7] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors and applications to invisible sanitizable signatures," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10175 LNCS, pp. 152–182, 2017, doi: 10.1007/978-3-662-54388-7_6.

[8] I. Puddu, E. Zurich, A. Dmitrienko, and S. Capkun, "μchain: How to Forget without Hard Forks," pp. 1–21, 2016, [Online]. Available: https://www.airbnb.com/.

[9] D. Deuber, B. Magri, and S. A. K. Thyagarajan, "Redactable blockchain in the permissionless setting," Proc. - IEEE Symp. Secur. Priv., vol. 2019-May, pp. 124–138, 2019, doi: 10.1109/SP.2019.00039.

[10] R. Matzutt et al., "A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10957 LNCS, pp. 420–438, 2018, doi: 10.1007/978-3-662-58387-6_23.

[11] A. Chepurnoy, M. Larangeira, and A. Ojiganov, "Rollerchain, a Blockchain With Safely Pruneable Full Blocks," 2016, [Online]. Available: http://arxiv.org/abs/1603.07926.

[12] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," Ethereum Proj. Yellow Pap., pp. 1–32, 2014.

[13] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Artif. Life, 2008.

[14] C. Dwork, M. Naor, and H. Wee, "Pebbling and proofs of work," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 3621 LNCS, pp. 37–54, 2006, doi: 10.1007/11535218_3.

[15] A. Back, "Hashcash - A Denial of Service Counter-Measure," Http://Www.Hashcash.Org/Papers/Hashcash.Pdf, no. August, pp. 1–10, 2002.

[16] A. V. and K. S. Sankar, "Study of Chunking Algorithm in Data Deduplication," Adv. Intell. Syst. Comput., vol. 398, pp. 319–329, 2016, doi: 10.1007/978-81-322-2674-1_2.

[17] M. S. Niaz and G. Saake, "Merkle hash tree based techniques for data integrity of outsourced data," CEUR Workshop Proc., vol. 1366, pp. 66–71, 2015.

[18] Y. Sasaki and K. Aoki, "Finding preimages in full MD5 faster than exhaustive search," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5479 LNCS, pp. 134–152, 2009, doi: 10.1007/978-3-642-01001-9_8.

[19] "General Data Protection Regulation (GDPR) – Official Legal Text." [Online]. Available: https://gdpr-info.eu/.

[20] "Protecting the 'right to be forgotten' in the age of blockchain." [Online]. Available: https://theconversation.com/protecting-the-right-to-be-forgotten-in-the-age-of-blockchain-104847.

[21] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," Nist Spec. Publ., vol. 145, p. 7, 2011. https://doi.org/10.1136/emj.2010.096966.

[22] L. Er-rajy, M. A. El Kiram, M. El Ghazouani, New Security Risk Value Estimate Method for Android Applications, The Computer Journal, Volume 63, Issue 4, April 2020, Pages 593–603, https://doi.org/10.1093/comjnl/bxz109

[23] Ikidid, A., and E.F. Abdelaziz. 2019. "Multi-Agent and Fuzzy Inference Based Framework for Urban Traffic Simulation." In Proceedings - 2019 4th International Conference on Systems of Collaboration, Big Data, Internet of Things and Security, SysCoBIoTS 2019. https://doi.org/10.1109/SysCoBIoTS48768.2019.9028016.

[24] Ikidid, Abdelouafi, and Abdelaziz El Fazziki. 2020. "Multi-Agent Based Traffic Light Management for Privileged Lane." 8th International Workshop on Simulation for Energy, Sustainable Development and Environment, SESDE 2020, 1–6. https://doi.org/10.46354/i3m.2020.sesde.001.

[25] N. Zubaidi, R. G. Pratama, and S. Al-Fatih, "Legal Perspective on Effectiveness of Pre-Work Cards for Indonesian People," Bestuur, vol. 8, no. 1, p. 9, 2020, https://doi.org/10.20961/bestuur.v8i1.42722