Integrity Management and Access Control of External Storages using Blockchain Technology

Hasan Mohammad Shahriar
Research and Development
Kona Software Lab
Dhaka, Bangladesh
h.m.shahriar@konasl.com

Muhammad Nur Yanhaona
Research and Development
Kona Software Lab
Dhaka, Bangladesh
nur.yanhaona@konasl.com

***Abstract*—The inherent limitation of blockchain technology for storing bulky and often confidential information such as files and images is a major obstacle for the technology's adoption as a platform for real-world applications that would otherwise be greatly benefited from the technology's other offerings. As an alternative, this paper presents a novel solution for reliable and secure integration of external document storage systems such as storage clouds and data centers with a blockchain network. In this approach, each document in the external storage is tagged with some blockchain smart contract that governs user access. Consequently, the responsibility of document access control, access fee processing, integrity insurance, and user authentication remains in the hands of the blockchain network and external storage systems focus only on storing and delivering documents efficiently. The solution's reliance on the blockchain technology for various control operations related to document access means greater trust and security for these operations. At the same time, this responsibility breakdown simplifies external document storage systems' design.**

***Index Terms*—peer-to-peer computing, distributed information systems, document handling**

I. Introduction

Since its inception in 2008 [13], the blockchain technology has gained widespread attention as a transformative technology that can revolutionize many industries [1]. Blockchain based digital currencies such as Bitcoin [13], Ether [19], and Ripple XRP [7] are already being considered viable alternatives to existing currencies in trade and commerce for their security and ease of transfer. Blockchain smart contracts [16] [19], on the other hand, have spawned innovative applications in many business and financial sectors due to their capacity of encoding the rules of interaction and ensuring their enforcement.

Central to the appeal of blockchain technology is its maintenance of a distributed ledger of transactions – called the blockchain – in a peer-to-peer network of autonomous and anonymous entities. In a blockchain network, all entities are even and none of them is trusted; still, the security and integrity of the transaction ledger can be guaranteed. This feat is achieved by a complete replication of information in all network participants where each participant validates and executes all transactions. As long as the majority of the network participants are honest, the outcome of the transactions, i.e., the state of the blockchain ledger can be trusted [14].

This research is funded by KONA Software Lab, Limited.

The blockchain technology's decentralization of trust through information and processing replication in a theoretically infinitely scalable peer-to-peer network is leading innovations and renovations in many application domains where trust and information security are key concerns. However, problem in one area in particular appears to be a major obstacle for blockchain based application adoption. This is the problem of document storage.

The blockchain technology is inherently unsuitable for storing bulky information such as files and media contents due to the networking and storage cost associated with their management. Peculiarities of blockchain ledger maintenance such as *blockchain reorg* [2] further complicates the situation by making direct integration of existing trusted storage solutions with a blockchain network difficult. Finally, public blockchain technologies find the continual preservation and integrity insurance requirement for trustworthy document storage in conflict with their blockchain transaction ledger maintenance incentive where participants are only being paid for extending the ledger of transactions[1] and they can join or leave the network at any time.

Nevertheless there are some blockchain based or blockchain inspired storage technologies such as Ethereum Swarm [17], Filecoin [10], Storj [18], and IPFS [5] already available. These solutions break down a user's file into a series or hierarchy of data chunks then distribute the chunks to the peer-to-peer network. On a broad level, some of these storage solutions are like traditional distributed hash tables [11] [9]. Some others are like peer-to-peer file sharing services such as the popular Bittorrent [15]. These solutions apply some bitcoin-like incentive mechanisms on top of these base technologies to motivate the network participants to retain and serve data chunks upon users' request.

The motivation for these solutions is that they protect the users from vendor locked-in and they offer an overall larger storage capacity compared to existing storage alternatives. However, blockchain based solutions have the common problem that the owner (or user) has to take the responsibility of ensuring persistence and integrity of his/her data in the blockchain by retaining document metadata and issuing peri-

[1] by mining transactions into new blocks

odic audits. Furthermore, despite the combined storage capacity being huge, the download bandwidth can be significantly low as the network peers may be running simple commodity hardware behind low-speed network connections. In addition, designing incentive mechanisms for long-term persistent of documents in a mining based blockchain network is difficult

Legacy databases of existing applications are also an obstacle for the applications' migration to the blockchain domain. Data stored in proprietary data centers are often confidential that a typical administrator may not be comfortable to put in the hands of anonymous blockchain participants. Further, when existing cloud storage providers [12] [3] have already solved the storage capacity, scalability, and cost-effectiveness problems for the clients; there is little motivation for moving data into a blockchain storage.

We believe, to steer blockchain application innovations, blockchain technology should be supportive of existing storage solutions instead of being their competitors. In other words, the goal should be integrating existing storage technologies with blockchain – not toppling them. A collaboration of technologies can bring the best of the both worlds. The blockchain technology can ensure integrity of external documents and control access to them according to the transparent governance of blockchain smart contracts and leave the actual storage, delivery and capacity scaling to a matured storage technology. Here, the blockchain technology is ideally suited for its part because information corruption in a blockchain network is very difficult and access rules written in the smart contracts are self-enforcing if integrated properly.

In our scheme, location, signature, access control configuration, upload/download fees and so on metadata information about the document is stored in blockchain smart contracts. A user gets access to the externally stored documents by interacting with a *Storage Integration Blockchain Gateway* from a blockchain client application. Any conversation with the gateway involves a series of transactions in the blockchain network and happens following the instructions of some secure interaction protocol. Finally, if the gateway approves the access request then it generates an access token to the external storage that the user uses to upload/download a document with the external storage directly. In case of a download, in particular, the client application verifies document authenticity by locally computing the document signature and matching it against the signature stored in the blockchain before delivering the document to the user.

Figure I depicts a high-level description of the system architecture of our solution. Observant readers will notice that the *Blockchain Network* and the back-end *Document Storage* of Figure I can scale up to meet users' high-availability and other quality of service needs. However, the same cannot be said about the *Gateway*. If proper care is not taken, its failure can make documents unavailable for client access. We avoided this grave potential problem by ensuring that the gateway database only contains information derived from the blockchain network. Hence, the gateway can be limitlessly replicated and the same back-end document storage may be

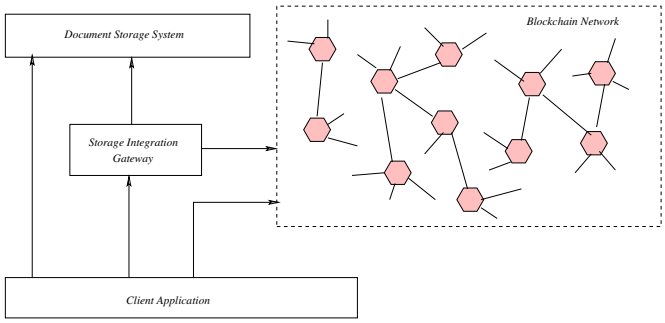

Figure 1. External Storage System Integration Model

interfaced by many gateways at the same time. Furthermore, we support different gateways to be configured differently to charge users for upload and/or download differently based on their quality of service (QoS) and application requirements.

The underlying core innovations that make our solution work are as follows:

1) Efficient, secure, and accountable document upload-download protocols that support configurable blockchain based payments.
2) A generic document access control configuration paradigm using blockchain smart contracts.
3) Enforcement of access control rules in the storage integration gateways.
4) Fault-tolerant design of the storage integration gateway against blockchain transaction reversal and back-end document storage failures.

This paper describes these core innovations and discusses some associated concerns. The rest of the paper is organized as follows:

A. Paper Organization

Section II elaborates on the scope of the external storage integration problem in our modeling, Section III describes document upload/download protocols and analyzes their characteristics, Section IV presents our innovation on blockchain smart contract based document access control policy configuration and its enforcement, Section V examines some gateway design concerns, Section VI discusses some related work on blockchain based/inspired document storage, Finally, Section VII concludes the paper.

II. The Scope of the Storage Integration Problem

To discuss the scope of the external storage integration problem, readers first need to understand our vision of responsibility breakdown for blockchain powered applications requiring document storage. For this latter discussion we refer to the model of Figure I.

We envision that the application logic of a blockchain powered application (subsequently referred as a *Blockchain Application*) will be stored in the blockchain network in the form of blockchain smart contracts. Information update, payment processing, and any auditing initiated by users' access

to the blockchain application will be governed by those smart contracts and reflected as transactions in the blockchain ledger. If we categorize aspects of users' interaction with a blockchain application into *4–As*: *Authentication* of user, *Authorization* of request, *Access* to information, and *Audit* of change; all *4-As* are handled by the blockchain network – except for the case of access/updating documents such as files and images.

Such documents will reside on one or more external document storage systems. At present, there is no convincing solution for reliable interaction of a blockchain network with an external information system in the literature. Hence *4-As* related to document access/update cannot be handled by the blockchain network as being done for other user interactions. A feasible alternative is that a user's blockchain identity and information in the blockchain ledger set the rules for the user's access to the external storage systems. Then a gateway service interacting with both the blockchain network and the storage systems enforces the *4-As* on behalf of the blockchain network.

To elaborate, a user will identify him/herself with the gateway with his/her blockchain identity and request upload/download of a document in an external storage as additional information associated with some blockchain smart contract. The gateway will check if the blockchain ledger contains permission information that support authorization of the user's request. Then the user and the gateway undergo an access authorization protocol that results in several blockchain transactions for auditing and payment processing. If authorization is successful, the gateway creates a restricted session with the external storage that the user uses to upload/download documents to/from the external storage.

The aforementioned breakdown of responsibilities for external storage access limits the scope of the storage integration problem to two primary activities:

1) development of a storage access permission control mechanism that exclusively uses blockchain information, and
2) designing secure, reliable, and accountable protocols for paid or free document upload/download with external storages.

Note that although we accept the blockchain network as the veritable source of information for both activities, the storage integration gateway needs to consider that any transaction in the blockchain can be reversed due to blockchain ledger reorganization [2]. Consequently, all access control decisions must be made based on the latest state of the blockchain ledger and the upload/download protocols should support rollback and resume. Doing this elegantly is a major design concern for the gateway.

Involvement of the blockchain network in document upload and download with external storages provides a natural mechanism for document integrity checking. During a document upload, a short and unique document signature can be generated from the document content[2] and stored in the associated blockchain smart contract. During a download, the client application can recompute the signature from the downloaded content and match that with the signature found in the blockchain smart contract. If the two signatures do not match then the document has been modified or corrupted outside the guidance of the blockchain network and the client rejects the document. This simple scheme of using blockchain ledger's immutability to ensure document integrity has been used by others also [4].

Subsequent sections chronologically describes the upload and download protocols, the permission control mechanism, and the gateway design.

[2]for example, a hash of the document byte stream can be the document signature

III. Document Upload and Download Protocols

For the discussion of this section, we ignore the access permission and blockchain ledger reorganization related concerns. Subsequent sections address those issues. In addition, we assume that both document upload and download with the external storage involves blockchain payments. Protocols without payment can be easily derived from the described protocols by eliminating the steps related to payments. Finally, both protocols heavily use symmetric key cryptography in various steps for information and payment security. We refer the uninitiated readers to [8] for a basic introduction to cryptography and to [6] for AES symmetric key cryptography in particular.

A. Quality Criteria for Upload/Download Protocol Design

We decide on a set of behavioral characteristics for the document upload and download protocols. These characteristics are classified into two groups based on the expectations of the storage integration gateway and that of the user from the protocols. From the gateway's perspective the following characteristics are important:

- Accountability: all actions should be traceable in the blockchain.
- Independence to storage system failure: payment is collected only after all interactions with the external storage are done successfully.
- Guaranteed Payment: the user can neither fool it to do unnecessary work nor withheld payment after the work is complete.
- Fault-tolerance: all local state information should be derivable from the blockchain ledger so that the gateway cannot corrupt any information and can be restarted easily after a failure.
- Security from Malicious Miners: no mining nodes can snatch the payment intended for the gateway.

From the user's perspective, on the other hand, the following characteristics are desired from the protocols:

- Security: no one else can interrupt the storage session intended for the user.
- Payment Security: the user must be able to to get refunds if he/she does not get the proper service.

- Confidentiality: the underlying document cannot be retrieved by others (users or miners) using the audit trail of the protocol execution in the blockchain ledger.

Propriety of both groups of behavioral characteristics is self-evident. Hence we do not elaborate on them further. During the protocol description, we discuss how different aspects of the protocols serve to achieve the mentioned characteristics.

B. Document Upload Protocol Description

The document upload protocol is composed of three main phases:

1) Token Generation: the *user* collects a payment token from the *gateway* and lock a document upload fee in the blockchain ledger for the *gateway*.
2) Document Upload Processing: the *gateway* verifies that payment is locked and adequate then cooperates with the *user* to upload the document in the *e*xternal storage.
3) Payment Finalization: the User verifies the document upload and unlocks the blockchain payment for the *gateway*.

Each phase of the protocol involves multiple interaction steps among various system components. Figure 2 presents the sequence diagram of the protocol. We now describe these steps as part of the three protocol phases.

1) Token Deposition: The *user* generates a unique document key D_k with document information (document name D_n, document uploader blockchain address DU_{addr} and document container contract address DC_{addr}) to identify each document.

$$D_k = hash(D_n, DU_{addr}, DC_{addr}) \tag{1}$$

The *user* requests the *gateway* to generate a token for uploading a document with document size D_s, document hash D_h, signature of document hash DH_{sig}, DU_{addr} and D_k. The *gateway* validates the input data from it user and verifies the signature using an already deployed smart contract (responsible to perform sign verification) from the blockchain node it is connected with. The *gateway* generates a AES symmetric key K_1 and encrypts its blockchain address G_{addr} with the generated key and produces an payment token M.

$$M = Enc(G_{addr}, K_1) \tag{2}$$

The *gateway* calculates the document upload fee according to the size of the document, stores K_1 for further use in next upload phase and returns the generated token M and calculated $fee\ amount$ to the User.

Next, the *user* generates another secret key K_2, encrypts the payment token M with the secret key and produces a upload token N.

$$N = Enc(M, K_2) \tag{3}$$

The *user* performs a transaction to the blockchain with token M and N and the document information (D_n, D_h and other document meta-data). During this operation, the calculated upload fee to upload the document is also deposited to the blockchain smart contract.

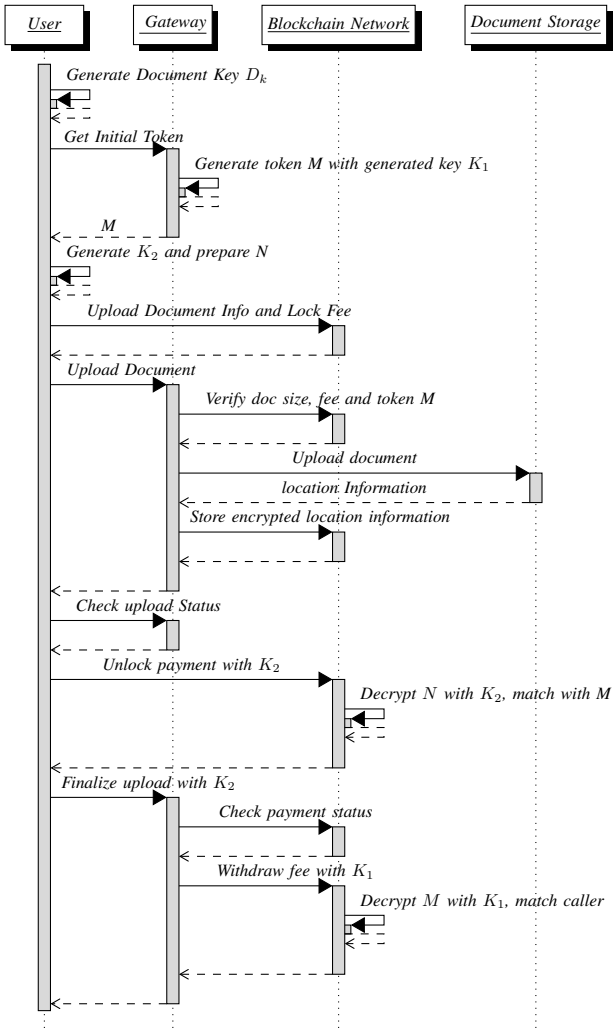

Figure 2. Sequence diagram of the document upload protocol

2) Document Upload Processing: After successful mining of the earlier transaction, the user sends the document upload request to the *gateway* with actual document and its meta-data. The *gateway* then retrieves the payment token M from the blockchain and verifies that it contains the *gateway's* address by performing a decryption on token M with key K_1 and checks that if the result is G_addr. The *gateway* also verifies the document size and deposited fee from the blockchain. If all verifications succeed, the *gateway* uploads the actual document to an external storage, locally stores the document location information, and store an encrypted version of the location information in the blockchain. The *user* can now verify that the document is successfully uploaded by checking the blockchain.

3) Payment Finalization: After a satisfactory verification of the outcome of upload, the user issues an unlock payment transaction with the user secret key K_2 to the blockchain. The blockchain verifies the key by decrypting N with provided K_2 and matches with stored payment token M. If the verification succeed, the smart contract unlocks the payment to forward the

amount to the receiver address encrypted as payment token M.

$$M = Dec(N, K_2) \tag{4}$$

The user requests the *gateway* to finalize the document upload procedure with document information, signature and secret key K_2. The *gateway* verifies the signature of the user and checks the payment unlock status from the blockchain. The *gateway* issues a transaction to collect the upload payment with its secret key K_1. The smart contract decrypts payment token M with K_1 and matches the result with the caller address C_addr. If the address matches with decrypted result, smart contract transfers the payment to the caller address.

$$C_{addr} = Dec(M, K_1) \tag{5}$$

Protocol Analysis: Since it is cryptographically hard to produce an alternative address and key pair that satisfy Equation 5, only the *gateway* can collect the payment. On the other hand, since payment is locked until someone supplies proper K_2 in the transaction evaluating Equation 4, which is only known to the user, the *user* ensures that the gateway cannot collect the payment until the user verifies that the document is successfully uploaded. Finally, since the document's location information in the external storage is recorded by the *gateway* in the blockchain in an encrypted form, none but the *gateway* can interpret this information for future reference. This simultaneously ensures information security and gateway fault-tolerance.

Note that so far we did not address the possibility that the upload protocol terminates halfway in the execution for some machine or network failure. This issue can be tackled easily by associating an expiry time with the payment locking transaction. If the protocol fails before the *gateway* uploads the document in the external storage system, the user can withdraw the payment after the expiry time. If the failure happens after the document upload in the external storage, then the *gateway* can never collect the payment as the user is no longer there to unlock the payment for it. Hence, the *gateway* simply removes the document from the external storage in such cases.

C. Document Download Protocol

The Document Download Protocol is used to download a document from the blockchain. The document download protocol is a combination of four main phases. The User collects a payment token from the *Access Controller* and deposits download fee to the blockchain for the *Access Controller*. The *Access Controller* collects a download session from the external storage, perform some cryptographic operations and returns an encrypted session to the user. The User checks the transaction from the blockchain, unlocks the download payment and initiates a fee transfer request to the *Access Controller*. The user will collect the encrypted session encryption key from the blockchain and regenerates the download session by performing cryptographic operations on it, and downloads the document from the external storage directly by using the session.

The working procedure of each phase can be subdivided into different steps. The required operations to perform different steps is described below.

1) Download Token Generation: The user requests to the *Access Controller* to generate a payment token for downloading a document with document key D_k, document hash D_h, user blockchain address U_{addr} and signature of document hash DH_{sig}. The *Access Controller* receives and validates the input data and verifies the signature DH_{sig} using D_h as message. The *Access Controller* generates a payment secret key K_a and encrypts its blockchain address AC_{addr} with the secret key and produces an payment token T_p.

$$T_p = Enc(AC_{addr}, K_a) \tag{7}$$

The *Access Controller* also calculates the download fee to download the document, stores the secret key K_a and returns the payment token T_p with calculated download fee amount to the user.

2) Prepare Download Session: After receiving the payment token, the user generates two secret keys K_c and K_u. K_c is the common secret, shared only with the *Access Controller*. The user generates an unlock token T_u by performing encryption operation on T_p with the key K_u.

$$T_u = Enc(T_p, K_u) \tag{8}$$

The user issues a download payment transaction to the *blockchain* with two tokens T_p and T_u, document information and deposits the payment amount. After successful mining of download payment transaction, the user issues a prepare download session request to the *Access Controller*. During this request the user sends document container contract address DC_{addr}, user address U_{addr}, document key D_k, document hash D_h, signed document key DK_{sig} and the K_c to the *Access Controller*. The *Access Controller* validates the input data, and also verifies the sign data and payment status from the *blockchain*. In return of successful verification of payment status, the *blockchain* returns the payment token T_p to the *Access Controller*. The *Access Controller* then verifies the T_p by performing decryption operation on it with K_a and matches the result with its own blockchain address.

$$AC_{addr} = Dec(T_p, K_a) \tag{9}$$

After a successful address verification the *Access Controller* retrieves the external storage information for the document and requests the *External Storage* for a session S to download the document. The *Access Controller* generates a secret key K_s, encrypts S with the secret key and produces an encrypted session ENC_s. It also encrypts the document key with the common secret K_c and produces an encrypted document key ENC_{dk}.

$$ENC_s = Enc(S, K_s) \tag{10}$$

$$ENC_{dk} = Enc(D_k, K_c) \tag{11}$$

After performing encryption operations the *Access Controller* prepare a hash of the encrypted session and issues

a transaction to the *Blockchain* with these hash value and ENC_{dk}. The *Access Controller* stores all the input data and returns the transaction hash and ENC_s to the user.

3) Get Encrypted Session: The user issues a transaction to the blockchain to unlock the payment with K_u, D_k and U_{addr}. The *Blockchain* unlocks the fee amount performing decryption on the unlock token T_u with the secret key K_u provided by the user and matches the result with the payment token T_p. The *Blockchain/* unlocks the payment amount to withdraw, if the verification of token satisfies the following condition.

$$T_p = Dec(T_u, K_u) \tag{12}$$

The user issues a request to the *Access Controller* to get the download session with D_k, DK_{sig} and U_{addr}. The *Access Controller* verifies the signature, checks payment unlock status from the *Blockchain*. After a successful verification, the *Access Controller* encrypts the session secret K_s with common secret K_c and produces an encrypted session encryption key ENC_{ks}.

$$ENC_{ks} = Enc(K_s, K_c) \tag{13}$$

The *Access Controller* issues a transaction to the *Blockchain* to withdraw the payment with the *Access Controller* secret K_a and ENC_{ks}. Blockchain verify the secret K_a by decrypting the payment token T_p with K_a and match the result with the caller address. If the decrypted result matches the caller address, blockchain transfers the download payment fee to the caller address. It also stores the ENC_{ks} with the download document information. The *Access Controller* returns the transaction hash to the user. After successful mining of the transaction, the User collects the encrypted session encryption key ENC_{ks} from the blockchain.

4) Download Document: The User first decrypts the encrypted session encryption key ENC_{ks} with the common secret K_c and retrieves the session secret key K_s, then decrypts the encrypted session ENC_s with K_s to retrieve the original session S.

$$K_s = Dec(Enc_{ks}, K_c) \tag{14}$$

$$S = Dec(Enc_s, K_s) \tag{15}$$

Now the User can get the document directly from the External Storage using the session S.

IV. Document Access Permission Control

V. Storage Integration Gateway Design

VI. Related Work

VII. Conclusion

References

[1] Break through with blockchain. https://www2.deloitte.com/us/en/pages/financial-services/articles/blockchain-series-deloitte-center-for-financial-services.html. Accessed: 2019-05-06.

[2] Chain reorganization. https://en.bitcoin.it/wiki/Chain_Reorganization. Accessed: 2019-02-12.

[3] Google cloud storage: Features and benefits. https://cloud.google.com/storage/features/. Accessed: 2019-05-15.

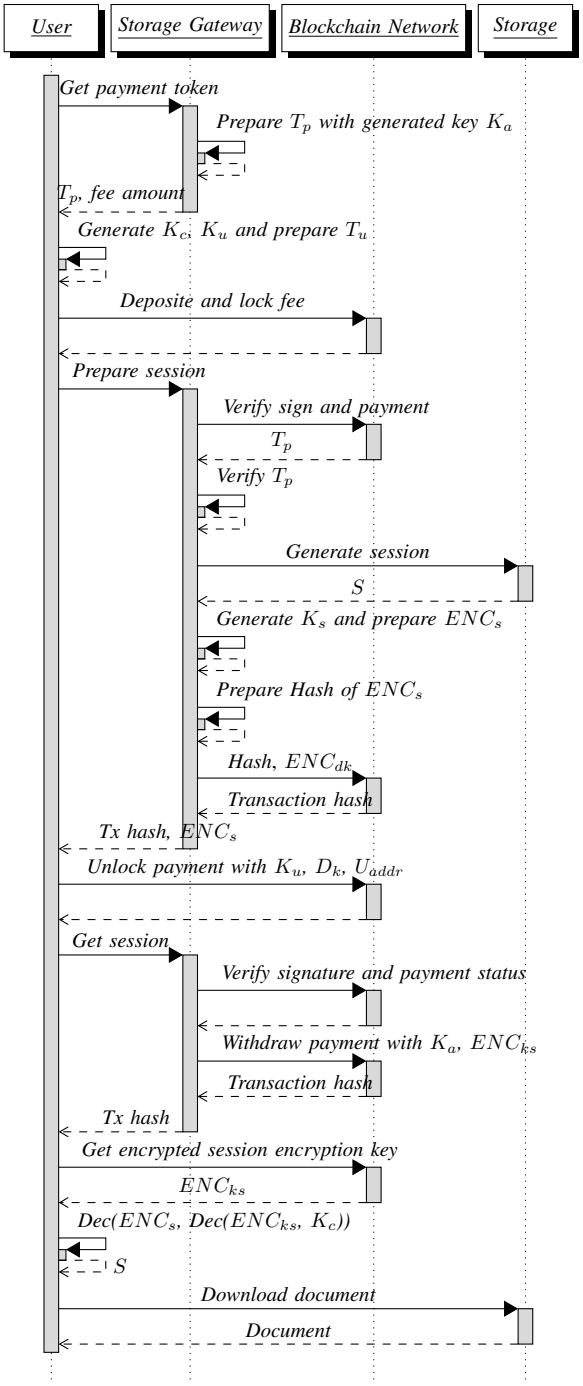

Figure 3. *Sequence diagram of download protocol*

[4] Seamless blockchain certification now even easier. https://stamp.io/. Accessed: 2019-05-24.

[5] Juan Benet. Ipfs - content addressed, versioned, p2p file system, 07 2014.

[6] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael, 1999.

[7] D. Stalin David. The ripple protocol consensus algorithm. 2014.

[8] W. Diffie and M. E. Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67(3):397–427, March 1979.

[9] Michael J. Freedman and David Maziéres. Sloppy hashing and self-organizing clusters. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II*, pages 45–55, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[10] Protocol Labs. Filecoin: A decentralized storage network. https://filecoin.io/filecoin.pdf, 2017.

[11] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.

[12] James Murty. *Programming Amazon Web Services*. O'Reilly, first edition, 2008.

[13] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[14] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 643–673, Cham, 2017. Springer International Publishing.

[15] Johan Pouwelse, PawełGarbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proceedings of the 4th International Conference on Peer-to-Peer Systems*, IPTPS'05, pages 205–216, Berlin, Heidelberg, 2005. Springer-Verlag.

[16] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[17] viktor trón, aron fischer, dániel a. nagy, zsolt felföldi, and nick johnson. swap, swear and swindle incentive system for swarm. https://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1, May 2016.

[18] Shawn Wilkinson, Tome Boshevski, Josh Brandoff, and Vitalik Buterin. Storj a peer-to-peer cloud storage network. https://storj.io/storj2014.pdf, 2014.

[19] D. Wood. Ethereum: a secure decentralised generalised transaction ledger. 2014.