Blockchain-based System for Secure Data Storage with Private Keyword Search

Hoang Giang Do
School Of Computer Science and Engineering
Nanyang Technological University
do0004ng@e.ntu.edu.sg

Wee Keong Ng
School Of Computer Science and Engineering
Nanyang Technological University
awkng@ntu.edu.sg

***Abstract*—Traditional cloud storage has relied almost exclusively on large storage providers, who act as trusted third parties to transfer and store data. This model poses a number of issues including data availability, high operational cost, and data security. In this paper, we introduce a system that leverages blockchain technology to provide a secure distributed data storage with keyword search service. The system allows the client to upload their data in encrypted form, distributes the data content to cloud nodes and ensures data availability using cryptographic techniques. It also provides the data owner a capability to grant permission for others to search on her data. Finally, the system supports private keyword search over the encrypted dataset.**

I. Introduction

In the past few years, there has been a significant growth in the interest to outsource data as well as operational services to clouds. Traditional cloud storage has come to rely almost exclusively on large storage providers acting as trusted third parties to transfer and store data. This system poses a number of shortcomings including the performance, availability, security, and high operational cost.

A decentralized cloud storage network has been introduced with many advantages over the datacenter-based storage. Similar to traditional solution, decentralized cloud storage network leverages client-side encryption to maintain data security. However, the management of encrypted data poses several challenges, the most important of which is data usability. More concretely, the data owner should have the capability to grant permission for others to search on the remotely encrypted dataset and obtain partial but useful content. A trivial solution is that the data owner retrieves back the whole data set, filters and sends to the authorized client the useful parts of data. However, that solution is infeasible due to the heavy cost at the client-side and it defeats the very purpose of outsourcing data.

In this paper, we present a system - BlockDS to address the aforementioned problem: authorized keyword searching on distributed data storage. The proposed system uses blockchain as the backbone for off-chain data storage access, permission grant and searches token generation.

II. Problem Formulation

System Model. Our proposed system involves three general parties: The data owner, data consumer, and the blockchain nodes as illustrated in Figure 1.

1) *Data Owner*. Data owner poses a data set of document. Each document contains a fixed set of keywords. Data owner wishes to outsource the dataset to save the cost of data storage and maintenance. At the same time, she also wants to be able to grant the permission for other clients to search on her outsourced dataset.
2) *Data Consumer*. A data consumer is a subscriber for the dataset provided by the data owner. If a data consumer subscribes for the dataset, he is granted the permission. He then is able to interact with a blockchain node to prove his credential and to perform keyword search on the outsourced data.
3) *Blockchain nodes*. They are the entities to maintain the blockchain. In our scenario, we consider separate cloud service providers as blockchain nodes and together forming a federated cloud.

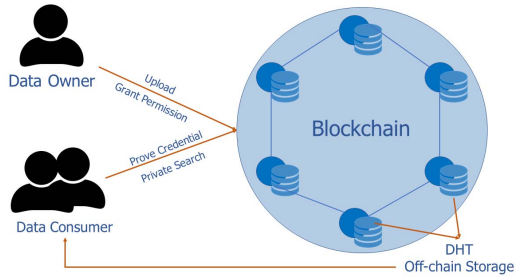

Fig. 1. BlockDS System Model

Workflow. To illustrate, consider the following example: A data owner Alice outsources her dataset of multiple documents to a federated cloud. Each document contains some certain keywords. In order to ensure data security, Alice applies data encryption at the client side before uploading to the clouds. The uploaded data include the encrypted documents and the encrypted keyword tags that facilitate the keyword search process. The encrypted dataset is stored distributedly by the cloud consortium, while the encrypted keyword tags are maintained by the blockchain. The encrypted keyword tags are used for indexing and providing a means to access the matching encrypted documents.

After the set-up phase, Alice is able to grant permission for a data consumer Bob to search on the encrypted dataset.

978-1-5386-2002-1/17 $31.00 © 2017 IEEE
DOI 10.1109/SERVICES.2017.23

Bob is now able to prove to the blockchain nodes that he has a particular permission for searching. Since there are usually multiple subscribers, the credential proof should be anonymous in order to preserve Bob's privacy for identity. More concretely, the blockchain can only verify that Bob is one of the subscribers, but can not extract Bob's identity from the proof. The blockchain nodes should also not be able to determine whether two particular proofs are created by the same person. Instead of having to download the whole dataset, Bob should have the capability to search for a keyword to retrieve only the necessary documents from the federated cloud.

III. Permissioned Blockchain Model

The emergence of blockchain technology is inextricably linked to the introduction of Bitcoin [1], the decentralized crypto-currency for the internet. Since its first introduction in 2008, blockchain has been successfully adopted in various important applications. The blockchain itself is composed of blocks, which sequentially link together by hash pointers. Thanks to the cryptographic features of digital hash function, the blocks are provably immutable. The blockchain itself does not depend on a central, trusted authority. Rather, it is distributed to all nodes participating in the network.

When Bitcoin introduced blockchain technology, it had public governance mechanisms. More specifically, anyone with a connection to the internet is able to access the blockchain data. In contrast to the permissionless settings, our system considers the scenario of a permissioned blockchain system. The entities having access to the blockchain are distinct cloud service providers with competing interests. This model provides various advantages over the permissionless model in terms of performance, security, and operational cost.

The permissioned blockchain model allows us to develop an access control layer for our system. The clients are able to access to blockchain content via the communication with one of the blockchain nodes. Different client roles have different levels of data access. Firstly, a data consumer is only allowed to access the content that is required to construct a credential proof. After being authenticated, the data consumer is granted to access to the keyword search component of the system. The component is another part of the data stored in the blockchain. Our keyword search component is modeled as a smart contract. A smart contract is a program that runs on the blockchain and has its correct execution enforced by the whole nodes in the network.

IV. BlockDS System

A. Overview

The proposed system BlockDS consists of three main components:

1) Distributed Data Storage. The component provides a decentralized cloud storage platform. The data are distributedly stored by the federated cloud. They are distinct organizations and can further distribute the data content. Proof of storage (with the use of blockchain) is applied in order to ensure the integrity.
2) Anonymous Access Control. The anonymous access control component allows the data consumer to anonymously convince the blockchain nodes of the possession of a certificate issued by the data owner. That means the blockchain nodes only know that the client's access is approved by the data owner, but they do not know who is the data consumer (there are multiple of entities of data consumer), and how many times he has shown the credential proof.
3) Private Keyword Search. The component provides an elegant mechanism for a data consumer to identify the specific encrypted data. Rather than downloading the whole data set, the data consumer interacts with blockchain nodes and the blockchain content to get the fingerprint of the specific data content based on the keywords (or the attributes). Finally, the data consumer retrieves the encrypted documents from the distributed off-chain data storage.

B. Distributed Encrypted Data Storage

BlockDS allows data owner to outsource a large amount of data to the system. Files should be encrypted at the client-side before being sent to the system. The system shards the data content and distributes the shards across peer nodes to reduce the impact of content delivery on any given node. Data are sufficiently sharded across the nodes and replicated to ensure high availability.

In order to maintain the access to the encrypted files, BlockDS leverages decentralized off-chain distributed hash-table (DHT) that is accessible through the blockchain, which stores references to the data but not the data themselves. DHTs have been widely used to coordinate and maintain metadata about peer-to-peer systems. Kademilia [2] is a popular DHT that allows efficient lookup through massive networks, low coordination overhead, and resistance to various attacks by preferring long-lived nodes. It has been widely used in peer-to-peer applications, including Gnutella and BitTorrent. Providing the fingerprint of a particular document to the DHT allows a client to correctly retrieve that file stored distributedly across the peer nodes.

The system leverages distributed proof-of-retrievability to maintain the integrity of the storage. We consider a particular file F consists of n data segments: $F = (F_1, F_2, \cdots, F_n)$. A basic proof-of-retrievability takes the form of a challenge-response protocol in which a node P demonstrates its possession of a file F and the fact that it can be correctly retrieved. To audit P's possession of F, P receives a random challenge c at regular basis; it produces a response r, which it can be publicly verified without possessing F. A basic proof-of-retrievability scheme consists of three protocols:

- $Setup(F) \rightarrow \{digest\}$. P computes a Merkle tree whose leaves are segments of the file F (with their indices) and whose root is $digest$. P outputs $digest$ value.

- $Prove(R) \rightarrow \{F_{r_i}, \pi_i)_{r_i \in R}\}$. $R = r_1, ..., r_k \in [n]$ denotes a set of random challenge received by node P. P outputs a proof that for each challenge index r_i in R, F contains F_{r_i} and the accompanying path π_{r_i} in the Merkle tree.
- $Verify(digest, R, F_{r_i}, \pi_i)_{r_i \in R}) \rightarrow \{0, 1\}$. The validation process verifies the Merkle path π_{r_i} for each segment F_{r_i} against the $digest$.

Proof-of-retrievability provides a strong guarantee, namely that with overwhelming probability, if P provides correct responses, F can be retrieved completely from P. we refer to the reader [3] for more details on the Proof-of-retrievability protocol. We make use of a random oracle (modeled by a hash function) to generate the random challenges with a fixed interval schedule. The blockchain nodes responsible for particular challenges are required to record the proof to the blockchain (i.e. Similar to Storj [4]). The correctness of the proof can be verified by every other nodes so that the integrity of the data storage is maintained.

C. Anonymous Access Control

Anonymous access control component allows a data consumer to obtain a credential from the data owner so that at some later points of time, he is able to construct a non-interactive proof for his credential. The blockchain nodes accept the request only if the attached proof is valid. The design of the component is inspired by the idea of Zerocoin [5].

We take the following example to understand the intuition of the anonymous access control component. Let consider there is a public (i.e everyone can access) physical bulletin board. To produce a new credential for the data consumer Bob, the data owner firstly generates a pseudonym S for Bob and commits S using a secure digital commitment scheme. The resulted commitment C can be opened by a random number r known by Bob. The data owner pins C to the bulletin board, there is a set $Sc = (C_1, C_2, \cdots, C_n)$ of commitments in the board. At a later point, Bob is able to prove possession of such credential by producing two statements in zero-knowledge:

- He knows a commitment $C \in Sc = (C_1, C_2, \cdots, C_n)$.
- He knows the opening r for the commitment.

In our system, the permissioned blockchain is used as the public bulletin board. Both the data owner and the data consumer are able to access to the public parts of the data stored on the blockchain. The public parts contain the commitments that we have described.

We now present a concrete construction using cryptographic accumulator proposed by Benaloh *et al.* [6], and later improved by Camenisch *et al.* [7]. The accumulator scheme [7] comprises of four algorithms:

- $AccumSetup(\lambda) \rightarrow params$. Generates two primes p, q, computes $N = pq$, sample $u \in QR_N$. Output (N, u) as the parameters.
- $Accumulate(C) \rightarrow A$. On a set of primes $C = \{c_1, \cdots, c_n\}$, outputs accumulator $A = u^{c_1 \cdots c_n} \ mod \ N$.
- $GenWitness(v, C) \rightarrow w$. Input a prime number $v \in C$, ouputs a witness $w = Accumulate(C - v)$.
- $AccVerify(w, v, A) \rightarrow \{0, 1\}$. Verifies $A = w^v \ mod \ N$

The security of the scheme based on the hardness of Strong RSA and Discrete Logarithm assumptions. We refer the reader to [7] for more details.

The description of the anonymous access control component consists of four algorithms:

1) $Setup(1^\lambda) \rightarrow params$. On the input parameter λ, run the algorithm $AccumSetup(1^\lambda)$ to obtain (N, u). Generate primes p, q such that $p = 2^w q + 1$ for $w \geq 1$. Let $\mathbb{G}$ be the subgroup of ${Z_q}^*$ and select two random genenrators g, h such that $\mathbb{G} = \langle g \rangle = \langle h \rangle$.
2) $GenCred(S, params) \rightarrow (c, skc)$. Given pseudonym $S \in Z_q^*$, select a random $r \in Z_q$ and compute $c \leftarrow g^S h^r$ such that c prime and $c \in [A, B]$, where $2 < A$ and $B < A^2$ (for more detail, refer [7]). Set $skc = r$ and output (c, skc), submit c to the the blockchain.
3) $ShowCred(params, S, c, skc, Sc) \rightarrow \pi_S$ Given data consumer pesudonym S, a credential c and its secret key skc, compute $A = Accumulate(params, Sc)$ and $w = GenWitness(params, c, Sc)$ and output the following proof of knowledge:

$$\pi_s = ZKSoK\{(c, w, r, S) : AccVerify((N, u), A, c, w) = 1 \wedge c = g^S h^r\}$$

4) $VerifyCred(params, \pi, Sc)$. Given a proof π_S, and the public set of credential Sc, first compute $A \rightarrow Accumulate(params, Sc)$, then verify that π_S is the the aforementioned proof of knowledge on c, Sc. If the proof verifies successfully, output 1, otherwise output 0.

The zero knowledge proof appears in step 3 of the scheme is a non-interactive proof that only requires one round of communication. Camenisch *et al.* [7] present an interactive zero-knowledge proof of knowledge that an accumulator contains a committed value. The construction of the non-interactive proof in step 3 leverages Fiat-Shamir transform on the interactive proof.

The $Setup$ algorithm is performed by the data owner to generate system parameters. Next, when the data consumer wishes to obtain a credential for data access, he sends a request to data owner together with his pseudonym S. At this point, data owner runs $GenCred$ routine on the input S to generate a digital commitment and its secret key skc.

When the data consumer wishes to show his credential, he first scans through the blockchain (i.e. the bulletin board) to obtain the set Sc consisting of all credential issued by the data owner. He then runs the $ShowCred$ routine to generate a credential proof, and broadcast it to the blockchain nodes for verification. The blockchain nodes also collect the set of credentials in the blockchain and validate the proof using the $VerifyCred$ algorithm. The credential certification is accepted if the last routine outputs 1.

The data consumer and the blockchain nodes are both required to compute $A = Accumulate(params, Sc)$ that requires linear scan the blockchain data. The complexity of

the protocol linear to the size of the number of registered data consumers.

D. Private Keyword Search

The component allows the data consumer to identify the specific encrypted data that he is interested in without the requirement of downloading the whole dataset. The meta-data (i.e. the fingerprint) of the encrypted documents are stored in the blockchain, and only the authorized clients are able to search via it. The authorization process is done by the anonymous access control layer presented in section IV-C.

Let consider the data owner outsources a set of encrypted documents. The documents are stored off-chain and can be accessed through a DHT as discussed in section IV-B. The access key is computed using a hash function (i.e. the fingerprint of the file). The blockchain does not store the actual data content, however, it maintains the access key data so that the data consumers are able to link to the DHT using blockchain.

We denote EKS as the encryption scheme that supports keyword search. The data owner appends to the access key a list of EKS ciphertext of each keyword and stores it in the permissoned blockchain. A document F with keywords $W_1, W_2, \cdots, W_n$ is stored in the blockchain under the structure: $H(F)||EKS(W_1)||\ldots||EKS(W_n)$. An authorized data consumer is able to produce a certain trapdoor T_w that enables a smart contract to test on each data entry whether one of the keywords associated with the access key (e.g. the document) is equal to the word W. Given a trapdoor and EKS ciphertext, the blockchain nodes can only test whether $W = W'$, and nothing else.

A typical keyword search cryptosystem consists of four general algorithms: *(1) KeyGen*: generates cryptosystem key; *(2) Trapdoor*: produces trapdoor T_W for a keyword W; *(3) Encrypt*: produces a EKS ciphertext for keyword W; *(4) Test*: tests whether keyword in the trapdoor is matched to the EKS cirphertext.

In our scenario, an additional algorithm is required for the data owner to produce a secret search key for the data consumer. We denote that algorithm *KeyDerive*. Three algorithms *KeyGen*, *Encrypt*, and *KeyDerive* are performed by the data owner, while *Trapdoor* is run by the data consumer to generate trapdoor, and finally, the *Test* algorithms is done by the smart contracts or the blockchain peers. We modify the protocol proposed by A. Popa *et al.* [8] to adapt our problem settings.

We start the protocol description by reviewing a few concepts related to bilinear maps. We will use the following notation: G_1 and G_2 are two (multiplicative) cyclic groups of prime order p, g_1 is a generator of G_1 and g_2 is a generator of G_2. A bilinear map is a map $e : G_1 \times G_2 \rightarrow G_T$ with the two following properties: $(1) Bilinear : \forall u \in G_1, v \in G_2$ and $a, b \in Z$, then $e(u^a, v^b) = e(u, v)^{ab}$, and $(2) Non-degenerate : e(g_1, g_2) \neq 1$.

We denote $H : \{0,1\}^* \rightarrow G_1$ and $H_2 : G_T \times G_T \rightarrow \{0,1\}^*$ be two random oracles, and g_1, g_2, g_T are respectively the generators of groups G_1, G_2, G_T. The private keyword search system consists of five algorithms as the follows:

- *KeyGen*: $k \leftarrow Z_p$.
- *KeyDerive(k,s)* : $k_s \leftarrow g^{k/s}$.
- *Trapdoor(w,s)*: $T_w \leftarrow e(H(w)^s, k_s)$.
- *Encrypt(k,w)*: Random $r \leftarrow G_T$. Output: $c = (r, H_2(r, e(H(w), g_w)^k)$.
- *Test*: Parse $c = (r, h)$. Test whether $H_2(r, tk) = h$.

The data owner generates a secret key k for keyword encryption EKS, and derives data consumers' search keys. Each data consumer poses a secret $s \in Z_p$, which can be generated by a mapping from his pseudonym with the data owner. Using k and s, the data owner computes a search key k_s for the data consumer so that later he can use it for trapdoor construction.

The correctness of the protocol follows the two equations: $tk = e(H(w)^s, g_2^{k/s}) = e(H(w), g_2)^k$, and $H_2(r, tk) = H_2(r, e(H(w), g_2)^k)$

The above scheme has data hiding and token hiding properties. Data hiding (privacy) requires that the semi-honest adversary is not able to distinguish between ciphertexts of two values not matched by some token. Token hiding (privacy) requires that the adversary cannot learn the keyword that one searches for. For the proof of security, we refer the the reader [8] for details of the proof. The complexity of the protocol is linear to the number of documents stored accross the distributed network.

V. Conclusion

We present the design of a system called BlockDS. The system enables outsourcing of data storage to a fluid distributed network of service providers. The system makes use of blockchain technology to enforce the data integrity via proof-of-retrievability scheme. While the client encryption is leveraged for data security, a private keyword search component is designed for searching on the encrypted dataset. The processes of anonymous credential grant, proving credential, and private keyword search are also performed with the help of blockchain. Future work should address more complex requirements such as credential revocation, boolean keyword search, etc.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf," 2008.

[2] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Peer-to-Peer Systems, First International Workshop, 2002*.

[3] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptology*, vol. 26, 2013.

[4] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014.

[5] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *2013 IEEE Symposium on Security and Privacy, 2013*.

[6] J. C. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques, EUROCRYPT, 1993*.

[7] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *22nd Annual International Cryptology Conference, 2002*.

[8] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," *IACR Cryptology ePrint Archive*, vol. 2013.