Meta-Key: A Secure Data-Sharing Protocol under Blockchain-Based Decentralised Storage Architecture

Yue Fu

School of Electronic and Computer Engineering, Peking University, China

Email: fuyuefyu@126.com

Abstract—**In this paper a secure data-sharing protocol under blockchain-based decentralised storage architecture is proposed, which fulfils users who need to share their encrypted data on-cloud. It implements a remote data-sharing mechanism that enables data owners to share their encrypted data to other users without revealing the original key. Nor do they have to download on-cloud data with re-encryption & re-uploading. Data security as well as efficiency are ensured by symmetric encryption, whose keys are encrypted by user's public key. Then, the key-ciphertext is recorded into a blockchain system so reliability and secrecy are ensured. When data are necessary to be shared, proxy re-encryption is adopted in order to generate new symmetric keys as well as corresponding ciphertexts dedicated for data-sharing so security of user's original key can be well-remained.**

I. Introduction

A. Backgrounds

Cloud-storage is becoming a highly commercialized industry in our day-to-day life. Present cloud-storage mode authorizes management of personal data with the help of third-party institutions. Due to trusting in cloud-service vendors, users don't care how their data are arranged in what form of storage devices/algorithms. However, reliability of such centralised storage architecture totally depends on reliability of single cloud-service provider. Users are unable to get their data when vendors stop their service. Moreover, such mechanism is unsafe in nature: information stored in third-party servers may be eavesdropped, stolen or destroyed by politic, technological or legal methods.

Decentralised cloud-storage benefits from its architecture as well as low-cost. As if during P2P downloading every node contributes its bandwidth to boost globally, it provides a decentralised key-data storage service without a third-party vendor: every node in the network can contribute its disk space to store data from other nodes. Each node can be either space demander or provider. Trust between users and service providers is no longer necessary so every client's private data must be encrypted before uploaded on-cloud.

Such decentralised storage system can be realised by the blockchain skill. Ciphertexts are distributed in unknown nodes in the network so security is ensured by randomness. Information of the location where ciphertexts are stored is recorded in a blockchain system that is maintained by all nodes. Special ciphertext can only be decrypted by user who keeps the corresponding private key so no central node can be attacked.

In this way, data-security and user-privacy are well-ensured. Such mechanism realises a true decentralised storage: data are stored in unknown nodes; user can access data any time he want; participants bear no additional burden.

However, data-sharing issue turns out to be a problem under blockchain-based storage architecture due to 2 reasons: 1, Failure of traditional data-sharing manners compatible to new architecture. 2, Lack of secure protocol in decryption key sharing process. In this paper, a feasible data-sharing mechanism that can work under blockchain-based decentralised storage architecture is proposed. Symmetric key for data decryption is uploaded to blockchain system as a part of metadata. User's asymmetric key-pairs are used only for metadata encryption, which avoids possible security problems that may exist in user's symmetric key management. Proxy re-encryption as well as a ciphertext transformation & restoring mechanism is proposed to solve security problems in symmetric key sharing under untrustworthy environment.

B. Contributions

The contributions of this paper are summarized as below:

1) **Meta-Key Mechanism.** Symmetric key for data decryption is uploaded to blockchain system as a part of metadata and is encrypted by user's asymmetric key-pairs. This efficiently introduces a secure data-sharing method under blockchain-based storage architecture.

2) **Key Transformation & Ciphertext Restoration.** Proxy re-encryption as well as ciphertext restoration mechanism are introduced in blockchain-based storage to solve security problems brought by untrusted-node environment. Through this mechanism, original keys and storage locations are not involved during the data-sharing process so security problems are solved.

3) **Collusion Attack Resistance.** Collusion attack resistance is an issue yet to improve in proxy re-encryption schemes. At the end of this paper we show the nature collusion attack-free property in our proxy re-encryption scheme.

C. Paper outline.

The rest of the paper is organized as follows. In section II, some preliminaries and related work will be introduced briefly. In section III, a meta-key mechanism is proposed to introduce feasible data-sharing scheme compatible with

blockchain-based cloud-storage architecture. In section IV, proxy re-encryption is implemented on symmetric key sharing between untrusted nodes to enhance security in data-sharing process. In section V, security performance of our protocol is analysed. Finally, the conclusion will be drawn in section VI.

II. PRELIMINARIES AND RELATED WORK

A. Bitcoin and Blockchain

Bitcoin [1] is a cryptocurrency and a digital payment system in the form of P2P, and transactions take place between users directly, without an intermediary. These transactions are verified by network nodes and recorded in a public distributed ledger called blockchain.

Blockchain is an emerging decentralized architecture and distributed computing paradigm underlying bitcoin and other cryptocurrencies. It is the core technology of the digital encrypted monetary system represented by bitcoin, which is a public ledger that records bitcoin transactions. The blockchain technology resolves two important issues that have long been faced in the field of digital cryptography: the double-spending problem and the Byzantine general problem [2].

The core advantage of the blockchain technology is to be decentralized with the application of data encryption, time-stamp, distributed consensus and economic incentive. It can achieve P2P transactions based on non-centralized credit and coordination, so as to solve the central institutions of the high cost, low efficiency and data storage security and other issues [3], which makes the blockchain technology not only successful in the field of digital encryption money, but also in the economic, financial and social systems also exist in a wide range of applications [4], thus bitcoin has recently attracted extensive attention from governments, financial institutions, high-tech enterprises and the capital markets.

Still, the development of blockchain technology is a challenge in future. There are problems and limitations in the security, efficiency, resources and game theory yet to be resolved [5]–[8].

B. Blockchain-Based Decentralised Storage

Blockchain skill can be directly applied on cloud-storage architecture. Every node can be either space demander or provider. Due to the possible huge volume of data, we place key metadata of those data into blockchain system rather than storing themselves. When any node requests to store data, it queries blockchain system to find out a feasible node location for storage and encrypts data yet to store and put them to the former location. Other key information are also recorded into blocks as metadata. When user nodes needs to query their data, they access the blockchian to findout their metadata and decrypt them by their private key to reveal locations of their data. Then, they download their data from the corresponding nodes.

Nowadays, many blockchain-based cloud-storage systems are coming to the fore, such as Storj [9], Enigma [10]. As an example, metadisk [11] is sub-project of Metadisk. It is a blockchain-based cloud-storage architecture. Every node can be either demander or provider of storage resource. Data are encrypted so unreadable to non-user nodes; hash of data is recorded into blocks so they are temper-resistant; data storage locations are recorded into blocks so user nodes can find out where they are; redundancy storage ensures data resist to single node failure. Under this architecture, neither additional server nor additional manual intervention is necessary. Nodes autonomy themselves using blockchain skill.

C. Proxy Re-Encryption

Proxy re-encryption is a cipher transformation scheme that is widely used in the context of data-sharing in cloud-environment. It was first proposed by Blaze et al. in 1998 [12]. Without revealing any information about key or plaintext, it allows a semi-trusted proxy to transfer Alice's ciphertext to Bob's ciphertext with the same plaintext. "Semi trusted" means the proxy will strictly execute the encryption steps as the algorithm. Ateniese et al. formalized it into strict definition and proposed a series of proxy re-encryption schemes. Application in distributed storage systems are also discussed. It is widely used in many fields such as mail filter [13], distributed file system management [14] and intellectual property protection [15].

Most proxy re-encryption schemes are based on algebraic structure of encryption algorithm such as RSA so they are designed for asymmetric cryptography. However, in some case, symmetric cryptography may be better choice for data encryption such as sharing and encryption of large-volume data. Unfortunately most symmetric encryption algorithm can't work with proxy re-encryption. Syalim et al. proposed a proxy re-encryption scheme that directly accommodates symmetric cryptography in 2011 [16]. It can be applied in cloud services as a powerful tool for different applications.

III. DATA-SHARING IN BLOCKCHAIN-BASED CLOUD-STORAGE

A. Blockchain as a metadata store

Blockchain is naturally a decentralised storage system that is maintained by all nodes in the network. Unfortunately, this leads to a problem known as blockchain bloat. Every node must store a copy of every transaction in the blockchain so it will soon expand to an unmanageable size when stored with large data. Hence, some key information of the data are extracted and stored into blocks rather than full data. It may include file hash, storage location, etc. Then, full data are stored into somewhere in a trivial cloud storage. Both data and metadata are encrypted by users [11]. Fig.1 indicates how it works.

However, data-sharing issue turns out to be a problem under blockchain-based storage architecture. Existing data-sharing schemes are based on centralised cloud-service providers. Data owners directly offer their keys to service providers. The data-sharing process totally run by service providers, which relies on trust from data owners to service providers. Still, in blockchain storage systems, such agent no longer

exist and every node is untrustworthy. Hence, existing data-sharing schemes can not be applied directly and a secure data-sharing scheme that can work under blockchain-based storage architecture is necessary.

Moreover, in blockchain-based storage, user's private key should be involved in encryption of either data or metadata. Data owner must provide his decryption key to share the data so they are kept in owner's hand. This brings problem to management and possible security loopholes. Besides, sharing of decryption keys may bring additional problems in untrusted-node-environment. We are proposing a series of policies as an attempt of solution to these problems.

B. Meta-key mechanism

Data-sharing is often executed in this manner: data are encrypted by owners. Ciphertext is transferred through common channels while encryption key is transferred through secure channels. When any user requests to share the data, data-owner encrypts his data and upload the data ciphertext to cloud-storage. Then, he apply that user's public key to encrypt the decryption key and send the key ciphertext to the user. User decrypt the key ciphertext by his private key to get the decryption key and then download the data ciphertext yet to decrypt.

Symmetric cryptography has advantages in these point so it's suitable to work in data-encryption of cloud-storage: efficient in encryption-decryption process, easy in algorithm, offers only one ciphertext and decryption key oriented to large number of users. According to the blockchain-based cloud-storage architecture, we store data in cloud-storage with symmetric encryption.

However, blockchain-based cloud-storage architecture is different from classical centralised cloud-storage. Hence, some revision is necessary for the sake of consistency. Still, we encrypt the symmetric decryption key by user's public key and generate a key ciphertext. As a part of metadata, key ciphertext is stored into blockchain system so is called meta-key. It is a natural combination of data-sharing in classical cloud-storage and blockchain-based storage so bears no additional changes. Fig.2 shows how our meta-key mechanism working compatible with blockchain-based cloud-storage architecture.

IV. KEY TRANSFORMATION AND CIPHERTEXT RESTORATION MECHANISM

A. Backgrounds

Due to the speciality of blockchain-based cloud storage, nodes are untrustworthy so the secrecy of storage is totally ensured by randomness of data locations. These locations are recorded into metadata whose decryption can only be executed by private key owners so they know where their data are. After data-sharing, the corresponding locations are revealed to data-receivers so information of these locations are no longer secure. Hence, we set dedicated nodes for data-sharing after applying a transportation of ciphertexts.

On the other hand, we don't want original encryption keys that the data-owner hold to be exposed directly to other users
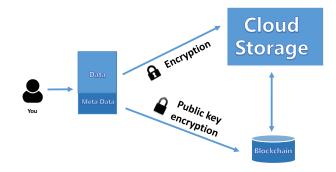

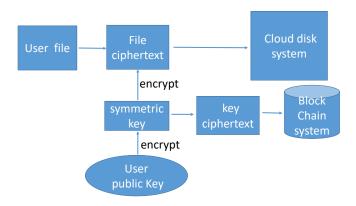Fig. 1. Blockchain-based cloud-storage architecture.

Fig. 2. Skeleton of meta-key mechanism, compatible with blockchain-based cloud-storage architecture

in the network. Instead, dedicated sharing keys are desired to set to corresponding data and user groups that can decrypt the same results(plaintexts) to data owners.

According to these needs, we designed a key-transformation & ciphertext restoration mechanism. Working with this mechanism, when any data owner is sharing his data to other users, only data location and a newly set decryption key should be provided. Data receivers download renewed ciphertext from the corresponding location and then decrypt it themselves.

The steps are summarized as below:

1) Let the original en/decryption key be S. Generate a random symmetric key S'.
2) Combine S and S' to create a transformation rule K.
3) Send a data-sharing request to a data storage node N_1. N_1 executes a ciphertext transformation according to K.
4) N_1 transports the transformed ciphertext to another node N_2 chosen by the data owner.
5) Data owner send S' as well as location of N_2 to data receivers. Receivers download the transformed ciphertext from N_2 and decrypt it by S'.

Through this mechanism, dedicated data-sharing nodes as well as decryption key are set without involving original locations of ciphertext and key S that data owners keep. The whole process is executed on-server without the need of ciphertext download & re-decryption & re-uploading.
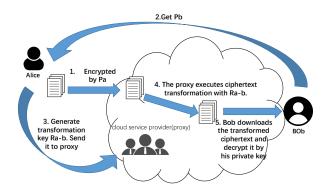
Fig. 3. Proxy re-encryption on-cloud.

Fig. 4. The final data-sharing process

B. Implement of Proxy Re-Encryption

To generate a transformation rule K, we need to find out a function that transforms a ciphertext from one key to another without having to decrypt or access the plaintext. For the sake of security, this K should be required to have such property: the user who owns the transformed decryption key S' can not calculate the previous key S even if he know all the transformed ciphertext. Meanwhile, the node who executes(knows) K knows nothing about S even if he know all the previous & transformed ciphertext.

Proxy re-encryption is a useful concept that can be applied in this context. In traditional cloud-service, it can be applied in this way: suppose Alice and Bob be 2 users of a cloud-service provider. the provider is not trusted by Alice so data of Alice is uploaded after encryption of Alice's public key P_a. Hence, the provider knows nothing about the plaintext. When Alice requests to share her data with Bob, she combine her private key and Bob's public key P_b to generate a transformation key R_k and send it to the cloud-service provider. Acting as a proxy, the provider executes the re-encryption with R_k. Hence, it's easy for Bob to download the re-encrypted ciphertext on-cloud and decrypt it by his private key. Obviously, this concept must fulfil the security demand we need in the last paragraph. The process is shown in Fig.3.

In our design, though a little difference is made, this concept still works. Let N_2 be the proxy and the data-owner be Alice whose en/decryption key is S. However, Alice generates S' herself rather than asking it from Bob. Then, she can calculate a R_k together with S and S', sending S' to Bob and R_k to the proxy.

Unfortunately, this can't work directly because most solutions for proxy re-encryption are designed for the public-key world. These schemes are based on algebraic structure like RSA hence can not be applied in our symmetric cryptography world. In practical use, the public-key proxy re-encryption can be efficiently used to re-encrypt the private key that is used to encrypt the data with the symmetric ciphers. However, this scheme has a weakness, because re-encrypting the key does not update the actual key used to encrypt data(the key of the symmetric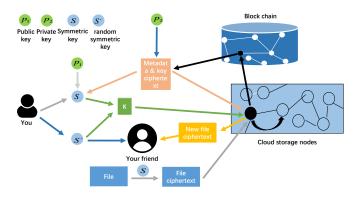 encryption) [16]. Hence, aiming at our goal, we need a proxy re-encryption scheme that can work directly in the symmetric encryption world.

C. Proxy Re-Encryption of Symmetric Cryptography

In 2011, Syalim et al. invented a symmetric cipher proxy re-encryption scheme using all or nothing transform [16]. Though designed for access control and different in encryption objects, the en/decryption algorithm can be directly adopted in our protocol for the proxy re-encryption step.

D. The Final Data-Sharing Process

As shown in Fig.4, the final data-sharing process are summarised into these steps:

1) Find out the metadata corresponding to user's data. Decrypt it with user's private key to get the decryption symmetric key S as well as data storage location N_1.
2) Generate a symmetric key S' randomly. Generate a re-encryption key K together with S and S'.
3) Send K to N_1 and let N_1 execute re-encryption with K.
4) N_1 sends the transformed ciphertext to another node N_2 randomly chose by the data owner for data-sharing.
5) Data owner share location of N_2 as well as S' to data receivers through a safe channel.
6) Data receivers download the renewed ciphertext from data-sharing nodes and decrypt it with S'.

V. SECURITY ANALYSIS

In this section, we are evaluating the security performance of our protocol by analysing potential attacks.

A. The Collusion Attack

Most of efficient proxy re-encryption schemes are not resistant from the collusion attack: with the knowledge of K, previous/transformed ciphertext and Bob's private key P_b', Alice's private key P_a' can be calculated. Hence, when the proxy colludes with Bob, P_a' as well as plaintext is revealed.

Fortunately, our protocol is well-resistant to the collusion attack because the proxy role is divided to N_1 and N_2. N_1 who knows K and the previous/transformed ciphertext doesn't know S'. N_2 who knows the transformed ciphertext doesn't know the previous ciphertext.

Obviously, the collusion between N_1 and Bob who keeps S' works, but it's actually impossible because Bob doesn't know the location of N_1. Still, the collusion between N_1, N_2 and Bob may happen. It's hard to realise due to these reasons:

1) N_2 is randomly chosen by the data owner and doesn't know the location of N_1.
2) N_1 doesn't know the importance of its ciphertext because it's location is known by neither N_1 nor Bob.
3) Bob who knows the importance of data can not communicate with N_1 through N_2.

B. Known Plaintext Attack and Ciphertxet Only Attack

These properties are ensured by the special proxy re-encryption scheme we chose to use. They are already proofed in [16].

C. Attack from Local Malwares

It's not convenient to encrypt all of a user's data by only one symmetric key. If several keys are used, key-management turns out to be a problem. If encryption keys are stored in user's hand, they may be eavesdropped, destroyed or stolen by malwares locally installed in user's computers.

In our protocol, encryption keys are no longer stored locally in owner's hand so it is easy for data owners to manage their encryption keys by only one private key. Moreover, one-time pad turns out to be possible if we store the encryption key flow into blocks. This brings a big promotion on security.

D. Reliability of metadata

Metadata is the most important part in our protocol without whom data may be lost forever. Due to the reliability of blockchain system, metadata are resist from single node failure and are always readable by those who owns the corresponding private key. On the other hand, blockchain system is highly retrospective. Hence, file storage and sharing record can be reliably stored and retrospect efficiently.

VI. CONCLUSION

In this paper, a secure data-sharing protocol under blockchain-based cloud-storage architecture is introduced. A meta-key mechanism compatible with existing architecture is introduced to arrange encryption keys with user's private key. Proxy re-encryption with some revision is applied to make data to be shared with high efficiency as well as enhancement of security. Details about security are discussed at the end of this paper, including collusion attack resistance property that doesn't exist in most practical proxy re-encryption schemes.

ACKNOWLEDGEMENT

REFERENCES

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Consulted, 2008.
[2] 19 Antonopoulos A M. Mastering bit-coin: Unlocking Digital Cryptocurrencies. USA: O'Reilly Media Inc, 2014.
[3] Swan M. Blockchain: Blueprint for a New Economy. USA: O'Reilly Media Inc, 2015.
[4] Technical report by the UK government chief scientiflc adviser [Online], available: https://www.gov.uk/government/uploads/system/uploads/attachment data/flle/492972/gs-16-1-distributed-ledger-technology.pdf, February 21, 2016
[5] Ethereum White Paper. A next-generation smart contract and decentralized application platform [Online],available:https://github.com/ethereum/wiki/wiki/WhitePaper, November 12, 2015
[6] Brito J, Shadab H, Castillo A. bit-coin flnancial regulation:securities, derivatives, prediction markets, and gambling.The Columbia Science & Technology Law Review, 2014, 16:144?221
[7] Eyal I, Efe Gencer A, Sirer E G, van Renesse R. bit-coin-NG:a scalable blockchain protocol. Cryptography and Security,arXiv: 1510.02037
[8] Courtois N T, Bahack L. On subversive miner strategies andblock withholding attack in bit-coin digital currency. Cryptography and Security, arXiv: 1402.1718
[9] Storj: A Peer-to-Peer Cloud Storage Network, https://storj.io/storj.pdf
[10] Zyskind G, Nathan O, Pentland A. Enigma: Decentralized Computation Platform with Guaranteed Privacy[J]. Computer Science, 2015.
[11] MetaDisk: Blockchain-Based Decentralized File Storage Application, http://metadisk.org/metadisk.pdf
[12] Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography[J]. Lecture Notes in Computer Science, 1998, 1403:127-144.
[13] Ateniese G, Fu K, Green M, et al. Improved proxy re-encryption schemes with applications to secure distributed storage[J]. Acm Transactions on Information & System Security, 2006, 9(1):1-30.
[14] Ibraimi L, Tang Q, Hartel P, et al. A type-and-identity-based proxy re-encryption scheme and its application in healthcare [A]. SDMc 08 [C]. Heidelberg: Springer, 2008:185-198.
[15] Taban G, Crdenas A A, Gligoret V D. Towards a secure and interoperable DRM architecture [A]. Proceedings of the ACM Workshop on Digital Rights Management 2006[C]. New York, USA:ACM, 2006: 69-78.
[16] Syalim A, Nishide T, Sakurai K. Realizing Proxy Re-encryption in the Symmetric World[M]// Informatics Engineering and Information Science. Springer Berlin Heidelberg, 2011:259-274.