Towards Blockchain-based Scalable and Trustworthy File Sharing

Shujie Cui, Muhammad Rizwan Asghar, and Giovanni Russello
Cyber Security Foundry
The University of Auckland
Auckland, New Zealand
Email: scui379@aucklanduni.ac.nz, {r.asghar,g.russello}@auckland.ac.nz

***Abstract*—In blockchain-based systems, malicious behaviour can be detected using auditable information in transactions managed by distributed ledgers. Besides cryptocurrency, blockchain technology has recently been used for other applications, such as file storage. However, most of existing blockchain-based file storage systems can not revoke a user efficiently when multiple users have access to the same file that is encrypted. Actually, they need to update file encryption keys and distribute new keys to remaining users, which significantly increases computation and bandwidth overheads. In this work, we propose a blockchain and proxy re-encryption based design for encrypted file sharing that brings a distributed access control and data management. By combining blockchain with proxy re-encryption, our approach not only ensures confidentiality and integrity of files, but also provides a scalable key management mechanism for file sharing among multiple users. Moreover, by storing encrypted files and related keys in a distributed way, our method can resist collusion attacks between revoked users and distributed proxies.**

I. Introduction

Blockchain technology, initially used in bitcoin [1], has become extremely popular in recent years. Compared with the centralised could computing platform, blockchain-based decentralised storage systems provide better security guarantees. In particular, malicious behaviours of involved parties can be detected using auditable information in transactions managed by distributed ledgers. Besides cryptocurrency, blockchain technology has recently been used for other applications, such as file storage [2]–[4].

Unfortunately, most of existing blockchain-based file storage systems can not revoke a user efficiently when multiple users have access to the same file that is encrypted. Actually, they need to update file encryption keys and distribute new keys to remaining users, which significantly increases computation and bandwidth overheads. For instance, Cai *et al.* [2] present a blockchain-based decentralised system to support encrypted keyword search over encrypted files, where all the files and keywords are encrypted with the keys shared among all the users. When a user is revoked, new keys need be generated and files need to be re-encrypted with those keys. In [3], Shafagh *et al.* present a blockchain-based system for sharing IoT data, where the proxy re-encryption technique [4] is used to manage the keys. This approach does not share the data encryption keys among authorised users. Whereas, each user needs a *transform key* to decrypt the data. When one user is revoked, the data owner has to generate and distribute the transform key for remaining users. In [5], Alansari *et al.* propose an access control system for cloud federations by combining the attribute-based encryption and Pedersen commitment primitives with blockchain technology. Their solution records users' attributes and access control policies in blockchain, which ensures integrity of the attributes and policies. However, the attribute-based encryption is not scalable when it comes to user revocation.

In this paper, we combine the blockchain with the proxy re-encryption technique proposed in [6], which is different from the one used in [3], and design a distributed file storage and sharing system. Our design not only ensures confidentiality and integrity of the file in presence of active malicious entities, but also provides a scalable key management method to support multiple users access. In particular, our method does not need any key updates or file re-encryption for user revocation. Moreover, by storing encrypted files and related keys in a distributed way, our method can resist collusion attacks between revoked users and distributed proxies or storage providers, which is not guaranteed in [6]. Furthermore, by recording the auditable information and access control policies in the blockchain, malicious activities can easily be detected.

II. Solution Overview

A. System Model

As shown in Fig. 1, our system consists of five different roles: data owner, data user, storage provider, re-encryption proxy, and miner. Each node in the system can play at least one role. In case of multiple roles, a node can not be a storage provider and a proxy at the same time. Moreover, the miner, storage provider, and re-encryption proxy manage a copy of the blockchain.

- **Data owner**: The data owner is the one who wants to share her files with others. She encrypts the files and upload them to a storage provider. For file sharing, she manages the keys and access control policies.
- **Data User**: Authorised data users can read and write shared files according to the access control policies. They can obtain files from storage providers by providing file identifiers. Specifically, we say the data users are writers when they update files, and they are readers when they just download files.

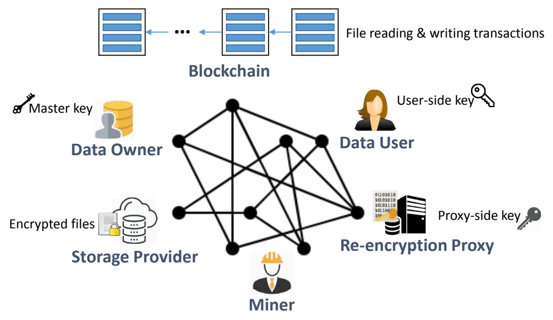

Fig. 1. Our system consists of data owners, data users, storage providers, re-encryption proxies and miners. The data owner uploads encrypted files to storage providers and uploads encrypted keys to proxies. Each authorised user has a unique user-side key; whereas, the proxy has a corresponding proxy-side key. The keys can only be recovered by decrypting the ciphertext with both the user-side and proxy-side keys. The blockchain includes file writing and reading transactions for auditing and verification as well as it contains access control polices to manage user access.

- **Storage Provider**: The storage provider stores the encrypted files and returns the files to authorised users when there is a request.
- **Re-encryption Proxy**: The re-encryption proxy assists authorised users to recover and update files. Specifically, it re-encrypts the data from writers to ensure other authorised users are still able to access, and it pre-decrypts the data for readers to ensure they can recover the file.
- **Miner**: The miner is responsible for verifying the file writing and reading transactions.

B. *Threat Model*

In our system, the data owner and user are trusted. However, we assume storage providers and re-encryption proxies are untrusted. On the one hand, they are interested in the stored files, and try to learn the content and abuse them for their benefits. External adversaries could also compromise the system to steal the files. On the other hand, they might delete part of the files and return dummy data to users for reducing storage, computation, and bandwidth overheads. Moreover, we assume attackers can not control more than 50% of the miners.

C. *Proposed Architecture*

In this section, we discuss high-level steps for uploading, sharing, and reading a file.

When uploading a file, the data owner first encrypts it with symmetric encryption, *e.g.,* AES-CBC, and sends it to a storage provider. Second, the data owner encrypts the symmetric key (hereafter, we call it a *secret*) with a *master key*, and sends the encrypted secret to the re-encryption proxy. Third, the data owner broadcasts a signed file writing transaction, which consists of identifiers, digests, and store locations of the encrypted file and secret. The miner will verify the transaction and add it to the blockchain if it is valid.

If the data owner wants to share the file with other users, for each authorised user, she first splits the master key into a pair of sub keys, where one sub key is sent to the user as the *user-side key*, and the other one is sent to the proxy as the *proxy-side key*. Only with one sub key, the user and the proxy can not decrypt anything. To recover the secret, the user needs the assistant of the proxy. Second, the data owner defines access control policies, signs them, and broadcast them as a transaction, which will be verified and added into the blockchain by miners. Moreover, in order to verify if the proxy pre-decrypts the encrypted secrets properly, the data owner also stores the digest of pre-decrypted secrets for each user.

When a user wants to access a file, it broadcasts a signed request, *i.e.,* the identifier of the required file. The miner first finds and checks corresponding access control policies of the requested file. If the user is authorised to access, the miner sends the store locations of the encrypted file and secret to the user. The user then sends the request to the related storage provider and proxy. After receiving the request, both the storage provider and the proxy will check the access control policies to see if the user is authorised to access. If yes, the storage provider will send the encrypted file to the user. The proxy decrypts the encrypted secret with the corresponding proxy-side key and sends it to the user. The user can recover the secret with her user-side key. Finally, the user can decrypt the file using the secret.

When a user is compromised, the data owner first updates the access control policies and generates a new transaction for the updated access control policies. Second, the data owner notifies the proxy to remove the user's proxy-side key. As a consequence, revoked users neither get the encrypted data from the storage provider, nor get the secret from the proxy.

III. Conclusion and Future Work

In this work, we present a secure file sharing system combining the blockchain with proxy re-encryption techniques. On the one side, our approach ensures confidentiality and integrity of outsourced files. On the other side, our approach provides a scalable key management mechanism for multiple access, where compromised users can be revoked efficiently without the requirement of updating the master key or updating users decryption keys. As for the future work, we will give the solution details and implement a prototype of the system to show its practical performance.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] C. Cai, X. Yuan, and C. Wang, "Towards trustworthy and private keyword search in encrypted decentralized storage," in *ICC 2017*. IEEE, 2017, pp. 1–7.
[3] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of IoT data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*. ACM, 2017, pp. 45–50.
[4] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.
[5] S. Alansari, F. Paci, and V. Sassone, "A distributed access control system for cloud federations," in *ICDCS 2017*, K. Lee and L. Liu, Eds. IEEE Computer Society, 2017, pp. 2131–2136.
[6] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *DBSec 2008*, ser. Lecture Notes in Computer Science, V. Atluri, Ed., vol. 5094. Springer, 2008, pp. 127–143.