2017 IEEE 31st International Conference on Advanced Information Networking and Applications

A Proposal of a Secure P2P-type Storage Scheme by using the Secret Sharing and the Blockchain

Masayuki Fukumitsu[*], Shingo Hasegawa[†], Jun-ya Iwazaki[‡], Masao Sakai[†], Daiki Takahashi[§]
[*]*Hokkaido Information University*
Faculty of Information Media
Email: fukumitsu@do-johodai.ac.jp
[†]*Tohoku University*
Center for Information Technology in Education,
Email: {hasegawa, sakai}@cite.tohoku.ac.jp
[‡]*Tohoku University*
Graduate School of Medicine,
Email: iwazaki@med.tohoku.ac.jp
[§]*Nissay Information Technology Co., Ltd.*

***Abstract*—We propose a new secure online storage scheme based on an open P2P network without a central server. In the proposed scheme, attackers cannot detect target user data in the online storage because the user data is divided into some parts by the secret sharing, and they are distributed to P2P nodes via the anonymous communication. The proposed scheme can prevent the direct attack on the target user node because metadata for the reconstruction also hidden in the online storage, hence nothing remains in the user node. Even if the state of the P2P network varies over time between the storing and restoring operation, the proposed scheme ensures that the user can identify the target nodes which stored his metadata by utilizing the Blockchain technology with only memorable secure information for user authentication. Furthermore, a malicious node to attack on others can be detected and ruled out by the mutual monitoring among the nodes and the majority decision rule.**

***Keywords*-Peer-to-Peer (P2P), Secret Sharing, Blockchain, Anonymous Communication.**

I. Introduction

The commercial online storage services such as Dropbox [1], Google Drive [2] and OneDrive [3] are becoming popular among current Internet users. These services are generally provided free of charge for most personal users. Instead of it, personal data stored in the online storages must be analyzed by the service providers to utilize the target advertisement. In addition, the providers might submit user's suspicious data to the national agencies such as the police for the criminal investigation without user's consent [4].

As a countermeasure against the invasion of user privacy, the user can encrypt his data before the upload to the online storage. However, it is not enough with just it. The providers and the agencies can patiently try to decode the encryption by the powerful offline bruteforce attack, and then eventually would steal the plain data due to the weak password problems [5], [6], [7] and the recent development of computer technology [8]. The use of the secret sharing [9] is a promising approach for preventing the offline bruteforce attack. The encrypted data in this case is divided into some parts (hereinafter called *shares*) by the secret sharing, and then they are separately distributed to the different online storages.

However, even the countermeasure using the secret sharing is not enough when you are targeted by a *strong* attacker which has an ability to widely eavesdrop on the Internet. The attacker can detect the target servers storing target user's share by eavesdropping the communication between the user and candidates for commercial online storage servers (whose number is not so large). Then, the attacker may be able to steal or delete the shares. If sufficient shares for the data reconstruction are stolen or deleted, the encrypted data is in danger of stealing or deleting. Moreover, the plain data also might be decrypted by the offline bruteforce attack.

In this paper, we propose a new P2P-type online storage scheme where the shares of the encrypted user data divided by the secret sharing are distributed to the randomly selected P2P nodes with the anonymous communication such as Tor [10] or I2P [11]. In the proposed scheme, the attacker cannot detect the target nodes storing the user's shares due to the difficulty of tracing communication between target user node and other nodes.

We here consider the case where the user restores his stored data from a conventional (typical) P2P-type online storage. In this case, the user requires key information for the reconstruction (hereinafter called *metadata*) such as locations of the target nodes storing his data and some passwords for decryption and user authentication, and so on. The metadata generally remains on the user node (i.e. user's local device), and the location of the target user's local device is obvious; therefore, the attacker can easily try to attack on it. To protect the stored data in the online storage, the metadata must be protected against the direct attack on the user node: that is, the metadata should be hidden from the user node.

In this paper, we propose a novel technique for hiding the metadata itself in the P2P network. In the proposed scheme, the metadata is encrypted, and divided into some shares, and then they are distributed to the seemingly random selected P2P nodes which are actually determined with the memorable authentication information such as user ID, password, and so on. Therefore, the only valid user can restore the metadata securely and completely from the P2P network with the authentication information, and his user data also can finally be restored with the metadata.

In this paper, we consider the pure P2P network. Namely, there is no central server which manages and controls the whole network. In addition, the P2P network is modeled as a open public network. Hence, each node's status (e.g. reliability, availability, and so on) are diverse and different, and then the constituent nodes of the P2P network vary with time. This implies that the state of the P2P network in the restoring operation is changed from that in the prior storing operation. To ensure the correct restoring of the stored shares in such a difficult situation, we employ time history record of the P2P network which are periodically and permanently generated. In each record (hereinafter called *storage node list*), available P2P nodes for storage server at the recording time are randomly listed and the time stamp is labeled. When the metadata is stored and restored, the user arbitrarily selects one of all the existing storage node lists, and then *deterministically* determines target nodes for storing and restoring the metadata's shares from the selected list with the user ID and password. Therefore, by using the same authentication information (i.e. a tuple of user ID, password, and a selected storage node list) between the storing and restoring operations, the user can correctly identify the same target nodes even though the state of the P2P network varies.

By contrast, note that the user requires to memorize the selected storage node list as well as the user ID and password, and delete it from the user node in order to prevent the direct attack on the user device. Because it is quite meaningless to remain the selected list (which is another metadata) instead of hiding the metadata. However, the information of the storage node list is too huge and complex for most users to memorize it. We solve this problem by utilizing the *Blockchain* technology which underlies the Bitcoin [12]. In the Bitcoin system, the Blockchain is used as the public record (anyone can read it) of all transactions. Because of the excellent robustness and transparency of the Blockchain, the Bitcoin transactions can be conducted securely. In the proposed scheme, the Blockchain is used as the public record of all the existing storage node lists; therefore, the user can get arbitrary storage node list from the Blockchain by using only the corresponding time stamp. The time stamp is easy to memorize because of a simple and short format such as *YYYYMMDDhhmmss*. Namely, authentication information in the proposed scheme is a tuple of user ID, password, and time stamp of the selected storage node list.

Even though the time stamp is a simple and short, it feels so difficult for some users to memorize it in addition to the conventional user ID and password. Fortunately, to strictly memorize the time stamp is not absolutely necessary for the reconstruction. Even if the user forgets the time stamp selected in the storing operation, the user can try the restoring operation by using all the existing storage node lists recorded on the Blockchain with the correct user ID and the password. The computation cost for the whole search is at most linear with the number of the existing storage node lists. Besides, if the user memorizes only a part of the selected time stamp (e.g. 201703????05??), the computation cost can be reduced because of limiting the search range. Namely, there is a tradeoff between the computation cost (time) for the restoring operation and the cost of precise memorizing the selected time stamp.

As described above, the user ID and password are used as essential secret key for the user authentication in the proposed scheme. However, most passwords used by people are generally weak. In major providers services, the problem of the weak password is solved by using the two-step authentication [13] which employs an auxiliary device such as a smartphone. In the proposed scheme, the memorable time stamp can be an alternative to such an auxiliary device. Namely, the proposed scheme could expect to achieve the high security which is almost equivalent with the service employing the two-step authentication without any auxiliary device.

STORJ [14], which is a similar approach to a P2P-type online storage utilizing the Blockchain, has been already developed. In the STORJ, the Blockchain is used as the public record of all user's encrypted metadata. Therefore, the attacker can read all the encrypted metadata, and try to decode it by the offline bruteforce attack. Furthermore, the STORJ prevents the online password guessing attack to gain unauthorized access with the central authentication server provided by the service provider. Namely, the attacker can attack on the record of the Blockchain and the central authentication server. On the other hand, in the proposed scheme, the Blockchain has no secret information such user's metadata, and a malicious node (attacker) to perform the online password guessing attack can be detected and ruled out by the mutual monitoring among P2P nodes and the majority decision rule without the central server. Therefore, the proposed scheme can disable the attacks on the record of the Blockchain and the central server.

This paper is organized as follows. We describe main components of the proposed scheme in Section 2. In Section 3, we give a detail of the proposed scheme. We discuss the security of the proposed scheme in Section 4. Section 5 is the conclusion.

II. Preliminaries

In this section, we describe the main components of the proposed scheme, such as the P2P nodes, the user accounts and the storage node list.

A. P2P nodes

In this paper, we do not consider the central server which manages and controls the whole P2P network. Namely, we consider the pure P2P-type online storage. Each of nodes in the P2P network has a pair of public and secret keys and the unique IP address and port number for communicating with other nodes. The node uses the hash value of its public key as its identifier (hereinafter called *node ID*). Each node can enable or disable, respectively, following service functions:

- List creation service: Creating a new storage node list on the Blockchain as a new block,
- Bulletin board service: Gathering broadcast information and publishing it to the public,
- Anonymous relay service: Relaying on anonymous communication among unspecified P2P nodes,
- Storage service: Storing and restoring data from unspecified P2P nodes.

B. User Account

The user account consists of a pair of user ID and password which are required to correctly select same nodes in the storing and restoring operations.

In the proposed scheme, all users can use arbitrary user ID, however nobody cannot checks the collision of them due to not existing central server. Therefore, if there is a match of the complete authentication information (i.e. the user ID, the password and the selected time stamp) among all current users, then the stored user data are wrongly restored or updated by different users with a small probability. In a practice, we can decrease the probability of such troubles caused by the user ID collision by using the unique identifier (e.g. email address, telephone number) as the user ID.

C. Storage Node List

The storage node list is a randomly ordered list of the information about the available storage nodes which are P2P nodes enabling the storage service function. The information about each of the listed storage nodes consists of its node ID, its IP address/port number, its public key and its signature signed by its secret key. When the number of candidates (i.e. current available storage nodes) is too large, the storage nodes in the list are preferentially selected in order of node's reliability from all the available storage nodes. The list is labeled by its created date and time (*time stamp*) as its identifier.

The storage node lists are created periodically and permanently. Each of the created lists is approved by a competition mechanism such as the Proof-of-Work [12] among the nodes enabling the list creation service (hereinafter called *list creating node*), and all the created lists record on the Blockchain. Because of the robustness and transparency of the Blockchain, all the created lists can be securely shared among P2P nodes without deleting and changing them.

Note that selected nodes and their order in each list can expected to be different from that of others due to the random selection. This difference increases the complexity (i.e. security) of the candidate combination of nodes for storing and restoring operation. Thus, it is desirable to create a new list as different as possible from the existing lists.

III. Proposed Scheme

In this section we describe the proposed P2P-type online storage scheme. Operations of the proposed scheme consist of the initialization for a new node, the information sharing among the nodes, the creation of a new storage node list, the anonymous communication, and the storing and restoring of the user data.

In the following, we assume that the P2P network consists of sufficiently many nodes and then the P2P storage keeps stable.

A. Initialization

In the initialization process, a new node generates a pair of public and secret keys randomly, and decides the IP address and port number for communicating with other nodes, and selects the enable services to offer to other nodes, as described in the subsection II-A.

B. Information Sharing

In the proposed scheme, the main control server of the P2P network does not exist. Thus the information sharing among the P2P nodes are generally achieved with broadcast communication. For example, each of the nodes broadcasts regularly its self-introduction information such as its node ID, its IP address and port number, its public key and its enable service functions, with its signature. This broadcasting teaches other nodes that the node is available. In addition, if a node detects the malicious node, the node also broadcasts the information about the malicious one.

By sending and receiving the broadcasting information each other, all nodes can share the information about the available nodes and the malicious nodes. The nodes enabling the bulletin board service archive such broadcasted information and continuously publish them. Namely even unstable nodes, which are not always online, can easily obtain the unreceived information broadcasted during offline through the bulletin board.

C. Creation of a New Storage Nodes List

We describe the methods of creating a new storage node list on the Blockchain. Each of the *list creating nodes* (the nodes enabling the list creation service) evaluates the reliability of each current available *storage node* by analyzing broadcasted information concerning its storage node on the bulletin board. Then each list creating node creates a unique list satisfying the condition of the subsection II-C.

After the creation, each node tries to record its created list on the Blockchain as a new block. The recording operation is competition against other nodes as the mining operation in the Bitcoin system [12]. Only one is approved by a mechanism such as the Proof-of-Work, and its list can be recorded on the Blockchain.

D. Anonymous Communication

In the proposed scheme, P2P nodes generally exchange data each other with the anonymous communication in order to prevent the eavesdropping by attackers. In the anonymous communication, data is sent from a original node to a designated terminal node by relaying it among several *relaying nodes* (the nodes enabling the anonymous relay service function). Moreover, each relaying node knows only the information about its previous and its next hop nodes because of the layered encryption at every hop. This implies that no one except itself can identify which node is the origin or the terminal of the anonymous communication.

Therefore the origin node can anonymously and secretly communicate with the terminal node. Moreover, due to the property of the anonymous communication, it is difficult for attackers to identify the terminal node even if they eavesdrop can the communication of the origin node. Namely, the use can protect the information about locations of storage nodes storing target user data against the attackers who eavesdrops the communication of the target user.

E. Storing Operation

The storing operation of user data is divided into two phases. In the former half, the user data is stored in the randomly selected P2P nodes, and then the metadata for restoring it remains. In the latter half, the metadata is stored in the deterministically selected P2P nodes with memorable authentication information. We describe these two phases respectively.

Storing of User Data:

1) Join a user device in the P2P network as a new node.
2) Fix the current user data M.
3) Encrypt M to the ciphertext C by using the randomly generated one-time password P^M_{otp}.
4) Divide C into the N shares $c_1, \ldots, c_N$ by using the secret sharing, where N is an arbitrary positive integer.
5) Get the information about the current available storage nodes from the bulletin boards.
6) From the information gotten in 5), select nodes $u^M_1, \ldots, u^M_N$ randomly, where the share c_i will be stored to the node u^M_i for $1 \leq i \leq N$.
7) Compute the hash value $H(c_i)$ of the share c_i, and then set the *restore key* as $k^M_i = H(c_i)$ for $1 \leq i \leq N$. The key k^M_i is the secret information corresponds the share c_i. Namely, the user can prove to the nodes u^M_i that he is the valid owner of the share c_i by showing the key k^M_i.
8) Send the key k^M_i to the storage node u^M_i for $1 \leq i \leq N$ with the anonymous communication, and inquire whether it is available respectively.

   If the storage node u^M_i already stores data corresponding to the same key k^M_i, then u^M_i rejects the inquiry. In this case, return to the step 6) and reselect the node u^M_i.

   On the other hand, if u^M_i does not store such data, then u^M_i accepts the inquiry. In this case, send also the share c_i with the anonymous communication and store the pair (c_i, k^M_i) in u^M_i for $1 \leq i \leq N$ respectively.
9) Set the metadata E, consisting of the one-time password P^M_{otp} and the pairs $(u^M_1, k^M_1), \ldots, (u^M_N, k^M_N)$.

Storing of Metadata:

1) Decide the pair of the user ID and password $(ID, PASS)$.
2) Get all the existing storage node lists $L(t_1), \ldots, L(t_\ell)$ form the Blockchain, where ℓ is the number of all the lists.
3) Select arbitrarily the storage node list $L(T)$ from all the lists.
4) Compute the hash value $P^E_{otp} = H(T, ID, PASS)$, which is the one-time password for encryption of the metadata E.
5) Encrypt E to the ciphertext Q by using the one-time password P^E_{otp}.
6) Divide Q into the J shares $d_1, \ldots, d_J$ by using the secret sharing, where N is an arbitrary positive integer.
7) Select the storage nodes $u^E_1, \ldots, u^E_J$ from the selected list $L(T)$ by some deterministic algorithm using $(T, ID, PASS)$. For example, select the node whose order in $L(T)$ corresponds to

   $H(j, ID, PASS, T) \bmod |L(T)| + 1$

   as u^E_i for $1 \leq j \leq J$.
8) Compute the hash value $H(j, ID, PASS)$ for $1 \leq j \leq J$, and then set the *restore key* for the shares d_j as $k^E_j = (T, H(j, ID, PASS))$.
9) Send the key k^E_j to the storage node u^E_j for $1 \leq j \leq J$ with the anonymous communication, and inquire

whether it is available respectively.

If there is even one node u_j^E which already stores data corresponding to the same key k_j^E, and then u_j^E rejects the inquiry. In this case, return to the step 2) and reselect the list $L(T)$.

On the other hand, if all the nodes u_j^E for $1 \leq j \leq J$ does not store such data, then all u_j^E accept the inquiry. In this case, send also the share d_j with the anonymous communication and store the pair (d_j, k_j^E) in u_j^E for $1 \leq j \leq J$ respectively.

10) Memorize only $(ID, PASS, T)$ and delete all information concerning the stored user data such as the metadata from the user device.

F. Restoring Operation

The restoring operation of user data is the reverse order of the storing one and divided into two phases in a similar manner to the storing one. In the former half, the metadata is restored from the deterministically selected P2P nodes with memorized authentication information. In the latter half, the user data is restored from the randomly selected P2P nodes with the metadata. We describe these two phases respectively.

Restoring of metadata:

1) Join a user device in the P2P network as a new node.
2) Get the information about the current available storage nodes from the bulletin board.
3) Remember the $(ID, PASS, T)$ which decided in the storing operation.
4) Get the storage node list $L(T)$ labeled as the remembered time stamp T form the Blockchain.
5) Select the storage nodes $u_1^E, \ldots, u_J^E$ from the list $L(T)$ by the deterministic algorithm using the remembered $(T, ID, PASS)$ as described in the step 7) of the phase *"Storing of Metadata"* which is in the subsection III-E.
6) Compute the *restore keys*
$k_j^E = (T, H(j, ID, PASS))$, for $1 \leq j \leq J$
to restore the shares d_j for the nodes u_j^E.
7) Send the restore key k_j^E to the storage node u_j^E for $1 \leq j \leq J$ with the non-anonymous communication, and inquire whether the share corresponding to k_j^E can return respectively.

If the storage node u_j^E stores the share d_j corresponding to the key k_j^E, then u_j^E accepts the inquiry and returns the d_j to the client node.

On the other hand, if u_j^E does not store such share, then u_j^E rejects the inquiry or returns a dummy share to the client node.

8) Try to reconstruct the ciphertext Q of the metadata from the shares restored in the step 7).

If the reconstruction of Q fails, then this restoring operation also fails. Otherwise, go to the next step.

9) Compute the hash value $P_{otp}^E = H(T, ID, PASS)$, which is the one-time password for decryption of the ciphertext Q.
10) Decrypt the metadata E from the ciphertext Q by using the one-time password P_{otp}^E.

Restoring of User Data:

1) Get the one-time password P_{otp}^M for decryption of the ciphertext C, and the pairs $(u_1^M, k_1^M), \ldots, (u_N^M, k_N^M)$ from the metadata E.
2) Sends the restore keys k_i^M to the storage nodes u_i^M for $1 \leq i \leq N$, and inquire whether the share corresponding to k_i^M can return respectively.

If the storage node u_i^M stores the share c_i corresponding to the key k_i^M, then u_i^M accepts the inquiry and returns the share c_i to the client node.

On the other hand, if u_i^M does not store such share, then u_i^M rejects the inquiry or returns a dummy share to the client node.

3) Try to reconstruct the ciphertext C of the user data M from the shares restored in the step 2).

If the reconstruction of C fails, then this restoring operation also fails. Otherwise, go to the next step.

4) Decrypt the user data M from the ciphertext C by using the one-time password P_{otp}^M.

If the decryption of M succeeds, then requests all the used storage nodes u_i^M, u_j^E to delete the shares c_i, d_j by showing the corresponding restore keys k_i^M, k_j^E for $1 \leq i \leq N$, $1 \leq j \leq J$.

IV. Security Evaluation

There are three targets which attackers aims to break the security of the proposed scheme. The first is the P2P network storing shares of the user data and the metadata. The second is the user client device in which the user data and the metadata are generated or restored. The third is the memorized secret information for the user authentication (i.e. user ID, password, a selected time stamp). In this section, we discuss the security of these three attacking targets.

A. Security of P2P network

In the proposed scheme, the user data and the metadata are encrypted by the secure one-time passwords and then divided into the some shares by the secret sharing, respectively. All the shares are stored in the P2P nodes separately. Therefore, an attacker need to succeed all the following four steps in order to steal the specified target user data from the P2P networks. The first is to detect target storage nodes storing his shares. The second is to steal the shares from the target nodes. The third is to reconstruct the his ciphertext from the shares. The fourth is to decrypt the ciphertext to his user data or metadata. We discuss the risks of each steps.

1) Risk of Detecting Storage Nodes: In the proposed scheme, since the storage nodes for storing shares of user data are randomly selected, the secrecy of the target nodes' location achieves the highest level. Thus it is hard for attackers without the metadata to identify the target storage nodes, which the nodes are storing the shares of his user data.

On the other hand, the storage nodes for storing shares of the metadata are deterministically determined with user ID, password and a time stamp of used storage node list. This means that the variety of the existing storage node lists derives the complexity of the candidate combination of the storage nodes for storing the shares of the metadata. Therefore, it is difficult for attackers without his user ID, his password and his selected time stamp to detect the target storage nodes, which the nodes are storing the shares of the his metadata.

In addition, P2P nodes generally exchange shares via the anonymous communication. As a result, to eavesdrop the target user's communication does not help the attacker to detect the storage nodes storing his shares.

2) Risk of Stealing Shares: To restore a specified share from the storage node storing it through the protocol, it is required to show the corresponding restore key which is an evidence of the valid ownership of it. The restore key of the share is a hash value of the share, then an attacker needs the share itself to compute the key due to the one-way property of hash functions. This means that it is impossible for the attacker to obtain target user's shares without the restore key of them.

The metadata (including the restore keys of the user data's shares) divides into some shares and hides in the P2P network as well as the user data. Thereby, the attacker needs to know the restore keys of the metadata's shares, and also to know the locations of the storage nodes storing the shares in order to obtain the metadata. The restore keys and the locations are deterministically determined with a secret information for the user authentication (user ID, password and a selected time stamp). Thus the attacker cannot obtain the metadata's shares without the secret information.

3) Risk of Reconstructing Ciphertext: We consider the risk that an attacker reconstructs the ciphertext of user data or metadata from shares when the attacker succeeds to obtain some shares. To reconstruct the ciphertext, the attacker has to obtain the sufficient shares more than the threshold of the secret sharing. Therefore, we can decrease the risk by setting the threshold large.

4) Risk of Decrypting Ciphertext: The user data is encrypted by the randomly generated one-time password. Thus it is hard for attackers to decrypt the ciphertext without the metadata including the one-time password.

The metadata is also encrypted by another one-time password which is the hash value of the user authentication information including the frequently varying time stamp. Thus it is also hard for the attackers to decrypt the ciphertext of the metadata without the user authentication information.

Let us consider the situation where the ciphertext is stolen by the attacker. In this situation, the attacker eventually may decrypt the ciphertext by offline bruteforce attack, even the strong password is used in the encryption. However, the proposed scheme is resistant to such a situation which the ciphertext of the metadata is stolen. In the proposed scheme, the user can disable the old metadata by changing the nodes storing user's shares and updating the metadata. Thus by updating the metadata periodically, we can decrease the risk of the offline bruteforce attack against the ciphertext.

From these above discussions, we can understand that the security of the user data in the P2P network is strictly protected by the four steps, then it is very hard for attackers to steal the user data from the P2P network.

B. Security of User Device

In this subsection, we discuss the risk of the steal or the cracking of the user device.

We first consider the case where the user device is stolen. It is not difficult for attackers to directly steal the target user device since the location of it is usually obvious. In this case, even the device is protected with user password, the password could be broken with high probability by using the powerful offline bruteforce attack [8] due to the weak password problem [5], [6], [7]. Thus it is considered that the attacker can obtain all data in the stolen user device. However only a pair of public and secret keys and the public information remains in the stolen user device. The pair of public and secret keys is independent of the user data and the metadata. Therefore, to steal the pair of public and secret keys does not affect the security of the user data stored in the P2P online storage. For the public information, since anyone (including the attacker) can obtain it by browsing the bulletin board, then to steal it also does not affect the security of the user data.

We next consider the case where the user device is cracked. The user data or the metadata is temporarily existed in the user device while the user device runs the storing/restoring operations. Therefore, if the user data or the metadata is existing at the crack timing, the attacker can steal them.

C. Security of Secret Authentication Information

In this subsection, we discuss the risk of attack on the memorized secret information for the user authentication such as user ID, password and a time stamp labeled in a used storage node list.

In the proposed scheme, even if the user forgets the time stamp, the user can try the restoring operation with only the correct user ID and the password by searching all the existing storage node lists recorded on the Blockchain. In that sense, the user ID and the password are essential and are more important than the time stamp, then the user requires to protect them more securely. However, the attacker may guess target user's user ID and password from his personal information. Moreover, strength of most passwords used by people is generally weak as previously mentioned. Thus only the user ID and the password could not offer sufficient security of the proposed scheme. In order to solve such problem, the proposed scheme uses the time stamp as an auxiliary secure information.

The selected time stamp is expected to be independent of the user's personal information. Moreover, the available time stamp is frequently updated at the timing of changing the used storage nodes, then all the old metadata became unavailable after the update. This means that an attacker requires to detect the correct time stamp by the online bruteforce attack until the next update, even though the attacker know the correct user ID and password of the target user.

In addition, the proposed scheme can detect and rule out the unallowable online bruteforce attack with the mutual monitoring among P2P nodes and the majority decision rule. In the proposed scheme, the restoring operation of the metadata runs with the non-anonymous encrypted communication (i.e. simple SSL/TLS), namely the client node need to send its IP address information to storage nodes.

In this situation, let us compare the unallowable online bruteforce attack by the attacker (who does not sure anything) and the allowable user authentication by the valid user (who forgets only the correct time stamp). The attacker needs to search not only the correct time stamp but also the correct user ID and password. In this search process, the attacker sends multiple inquiries including conflicted restore keys such as $k_i^{E'} = (T, H(i, ID', PASS'))$ and $k_i^{E''} = (T, H(i, ID'', PASS''))$ to the same node. Note that these conflicted restore keys consists of different user ID and password while consisting the same time stamp T. These conflicted inquiries are inevitable because the attacker does not the correct user ID and password. On the other hand, the valid user, who know them, never send such conflicted inquiries to same node, in principle. The valid user send only one inquiry to one node at searching each one of candidate time stamps (but with a few exceptions by errors such as miss type). Therefore, if the storage node continually receives such conflicted inquiries from the client node using same IP address, the storage node can identify the client node as the attacker node and broadcasts the malicious IP address to other P2P nodes. Consequently, the online bruteforce attack using a few IP addresses can be ruled out by the information sharing and the majority decision rule.

V. Conclusion

In this paper, we have proposed a new secure online storage scheme based on a time varying open P2P network without the central server. The proposed scheme employs the encryption, the secret sharing, and the anonymous communication in order to store the user data securely in the P2P network. In the proposed scheme, in order to ensure the correct restoring of the user data from the time varying P2P network, the storage node list have been introduced as an auxiliary secure information in addition to the essential user ID and password. The storage node list, which is the randomly ordered list of the current available server nodes in the P2P network at a specific moment, are periodically and permanently generated in the continuous time. Since the time history record of the storage node lists needs to be shared with P2P nodes certainly, we apply the Blockchain technology to record them. Because of the excellent robustness and transparency of the Blockchain, the storage node lists can be preserved permanently without deleting and changing them.

We have discussed the security of the proposed scheme in a qualitative manner. The proposed scheme is resistant to variety of attacks except the cracking of the user client device during the storing/restoring operation is running. The security evaluation of the proposed scheme in a quantitative manner is our future work.

Acknowledgment

A part of this work is supported by JSPS KAKENHI Grant Number 16K00181.

References

[1] Dropbox, https://www.dropbox.com/, [Online; accessed Oct. 28th, 2016].

[2] Google Drive, https://www.google.co.jp/intl/ja/drive/, [Online; accessed Oct. 28th, 2016].

[3] OneDrive, https://onedrive.live.com/about/ja-jp/, [Online; accessed Oct. 28th, 2016].

[4] L. Kelion, "Microsoft tip leads to child porn arrest in Pennsylvania," BBC News, http://www.bbc.com/news/technology-28682686, 2014, [Online; accessed Oct. 28th, 2016].

[5] CSID:Consumer Survey: Password Habits, http://www.csid.com/wp-content/uploads/2012/09/CS_PasswordSurvey_FullReport_FINAL.pdf, 2012, [Online; accessed Oct. 28th, 2016].

[6] P. Ducklin, "Anatomy of a Password disaster - Adobe's giant-sized Cryptographic Blunder," https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/, Naked Security, 2013, [Online; accessed Oct. 28th, 2016].

[7] Z. Lin, "A Large-Scale Study of Web Password Habits of Chinese Network Users," Journal of Software, vol. 9, no. 2, pp. 293-297, 2014.

[8] D. Goodin, "25-GPU Cluster Cracks Every Standard Windows Password in <6 Hours," http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/, 2012, [Online; accessed Oct. 28th, 2016].

[9] A. Shamir, "How to Share a Secret," Commun. ACM vol. 22, pp. 612–613, 1979.

[10] Tor Project, Anonymity Online, https://www.torproject.org/, [Online; accessed Oct. 28th, 2016].

[11] The Invisible Internet Project, https://geti2p.net/ja/links, [Online; accessed Oct. 28th, 2016].

[12] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008, [Online; accessed Oct. 28th, 2016].

[13] Google 2-step verification, https://www.google.co.jp/intl/ja/landing/2step/, [Online; accessed Oct. 28th, 2016].

[14] Storj, A Peer-to-Peer Cloud Storage Network, https://storj.io/storj.pdf, [Online; accessed Oct. 28th, 2016].