JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 127-143 (2020)
DOI: 10.6688/JISE.202001_36(1).0008

A Cooperative Game-Theoretic Model in Cloud Storage Auditing

CHUNG-YI LIN AND WEN-GUEY TZENG
Department of Computer Science
National Chiao Tung University
Hsinchu, 30010 Taiwan
E-mail: {cylin.cs99g; wgtzeng}@nctu.edu.tw

In the traditional cloud storage auditing, users individually audit the cloud storage provider (CSP). However, users may pay the redundant auditing cost when they have similar auditing results. For example, a lot of users may get fail results when the CSP's service occurs accidents. In the worst case, the overall auditing cost linearly increases with the number of users. We propose a cooperative auditing model to let user share their auditing results in a friendship-based social network so that their overall auditing cost can be reduced. Furthermore, we design an auditing coalition game based on cooperative game theory, where not only users' individual interest but their entire benefit is c onsidered. We present two algorithms to find out an optimal way of forming auditing c oalitions. The first algorithm determines an optimal coalition in one iteration. In the second algorithm, after iteratively adjusting users' trust relationships, users may change to join another coalition if they receive wrong auditing results. The results of the one-iteration experiment show that the more number of users, the more auditing cost reduction. Specifically, the auditing cost can be reduced by 96% with respect to the original non-cooperative auditing model. In the multi-iteration experiment, the accuracy of users' auditing results can be improved from 88% to 100%.

Keywords: cloud storage auditing, data integrity, provable data possession, proof of retrievability, cooperative game theory, optimal coalition structure

1. INTRODUCTION

1.1 Problem Statement

Outsourcing computing operations to clouds has many advantages, such as ease of management, cost-effectiveness, lower impact outages, disaster preparedness, and simplified planning [1]. In these advantages, the cost reduction is a critical factor in the adoption of cloud storage. Several studies [2, 3] consider the difference of cost between buying local disks and leasing the storage from cloud services in their decision model. According to their results, users should rent the cloud storage to store their data if they spend lower cost on it.

A risk of storing data in the cloud storage provider (CSP) is that users may lose their data. Users' data may be incomplete if they are not reliably managed and stored by the CSP. Cloud storage auditing mechanisms, such as provable data possession (PDP) and proof of retrievability (POR) schemes [4, 5, 6, 7, 8], are proposed to remotely check integrity of data. To determine whether the risk occurs, a user can periodically verify the

Received April 6, 2019; accepted March 20, 2019.
Communicated by Tony Quek.

service quality of the CSP by using these mechanisms. If the CSP passes the check, the user can make sure that his data are well stored without corruptions. Otherwise, if errors occur, the user can request a compensation from the CSP for his loss. For example, in Amazon S3 service level agreement, if the user finds that the average error rate in a month is more than 1%, he can request a compensation by 25% of his payment.

However, in the existing auditing protocols, users individually check their own data. For most users, the cost that they spend on their individual works may be redundant. When the reliability of CSP's service is high, almost all users would receive pass results in their integrity checks. On the other hand, when CSP's service is corrupted in the foundation level, which may due to hardware failures, software bugs, configuration problems, and attacks, many users would get fail results. In a realistic case, a lot of websites and apps were affected by the outage of Amazon S3 service on February 28, 2017 [9]. Many users have the high error rates in accessing Amazon S3 at that time. Thus, when some users finish the auditing processes, the others' works become redundant because they would receive similar auditing results.

The problem is significant because the amount of users' auditing costs would be considerable. A user may spend cost on communication and computation when he audits. Furthermore, in order to let the auditing results be credible for the CSP, the user may authorize a trusted third party to perform the auditing work. In this case, the user must cover the auditing cost between him and the third party. When the number of users grows up, the overall auditing cost may become enormous.

1.2 Research Purpose

The purpose of this research is to let users cooperatively audit the CSP so that their redundant works can be reduced at lower auditing cost on average. In this cooperative model, some users choose to audit and share their auditing results in a social network. When the other users receive the auditing results, they can use them and choose to not audit. For two users, the criteria for using the auditing results from each other are based on their friend relationship and the strength of trust between them. The trust strength depends on the difference of users' opinions about the CSP's service quality. In particular, two friends can trust the auditing results from each other if their opinions about the CSP are similar.

However, in the simple cooperative model, users may face a problem that no one would like to audit if they consider their self-interest only. For example, there are three friends a, b, and c in a social network. a would like to use b's auditing results and chooses to not audit. Similarly, b and c prefer to request auditing results from c and a, respectively. As a result, no user would like to audit because their preferences are in a contradiction.

Cooperative game theory [10] is the approach to analyze the overall benefit gained by all users. Not only users' individual interest but their entire benefit would be considered to achieve a situation good for everyone. By using cooperative game theory, we motivate users to audit for optimizing the entire benefit without getting into conflict with their self-interest.

1.3 Main Contribution

We design an auditing coalition game to avoid users paying redundant cost on their auditing works. To our knowledge, our study is the first work to apply cooperative game theory to reduce overall users' auditing cost. In the game model, groups of users, who have the similar opinions about the CSP's service, form coalitions to cooperatively audit. In an auditing coalition, a user is elected as an *opinion leader* to perform the auditing task.

The rest users in the group use the leader's auditing result as a reference rather than audit by themselves. After playing the game, all users are assigned into auditing coalitions, where the overall auditing cost is minimal. Furthermore, every user has no incentive to leave his assigned coalition because his individual auditing cost would be the lowest. The set of users' coalitions in this situation is called an *optimal coalition structure*.

In our game model, we propose two algorithms to obtain the optimal coalition structure. The first algorithm finishes and generates an optimal coalition structure in one iteration. However, in an auditing coalition, users may receive wrong auditing results from the leader. For example, a user may receive a pass auditing result but find errors in his data when he uses them. The second algorithm is extended from the first in multi-iteration. In the algorithm, the trust strengths between users are iteratively adjusted. Users may join another coalition in the next iteration if they receive wrong auditing results.

In the experiments, we consider Erdős-Rényi (ER) random graph model [11] to simulate loosely-coupled and tightly-coupled friend graphs by $p = 0.3$ and $p = 0.8$, respectively. The results of the one-iteration experiment show that the overall auditing cost can be reduced by 90% for the loosely-coupled friend graph and 96% for tightly-coupled friend graph, respectively. The more friend relationships users build, the more auditing cost reduction. In the results of the multi-iteration experiment, the accuracy of users' auditing works is improved from 88% to 100% in the last iteration.

2. RELATED WORK

Ateniese $et\ al.$ [4] introduced the provable data possession (PDP) scheme, which allows users to remotely verify their stored data in an untrusted server without retrieving them. In their scheme, a user generates a metadata for a file. The metadata is used later for verification purpose. Barsoum $et\ al.$ [5] proposed a map-based provable multi-copy dynamic data possession (MB-PMDDP) scheme in cloud storage auditing. The users can validate the number and integrity of data replicas to audit whether the CSP meets the promise of replicas in the service level agreement (SLA). Lin $et\ al.$ [6] developed a mobile PDP (MPDP) scheme in mobile cloud computing. The mobile device users shift the computational burden of the PDP scheme to a trusted third party. The trusted third party can verify users' data without accessing contents of the file.

Zhu $et\ al.$ [12] proposed the cooperative PDP (CPDP) scheme by using hash index hierarchy. In this scheme, multiple CSPs cooperate to store and maintain user's data. When a user performs the auditing work for all CSPs, he uses a single value to verify the integrity of his data rather than separately audits them many times. The experiment results show that the user has lower computation and communication overheads when he audits his data in cooperative CSPs.

Juels $et\ al.$ [7] explored a proof of retrievability (POR) scheme to let users ensure the integrity of files when they retrieve them. Due to the encoding of the data file, $e.g.$, erasure codes, a user's entire file can be reconstructed from responses of the CSP.

Wang $et\ al.$ [13, 14] proposed a privacy-preserving public auditing protocol. A user can authorize a third party auditor (TPA) to verify the integrity of his data without revealing any knowledge on the data content. Liu $et\ al.$ [15] enhanced the TPA scheme to support variable-size blocks in dynamic data update. Li $et\ al.$ [16] developed a new TPA scheme with lightweight computations for the low-performance end device. The simulation results show that their protocol is about 300 times more efficient than the original protocol in [13]. Li $et\ al.$ [8] proposed a novel POR scheme specific to cloud

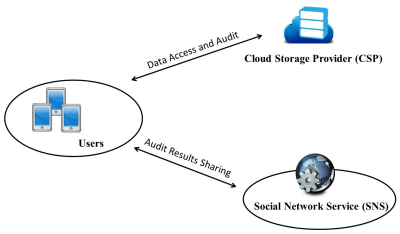

Fig. 1. System model of cooperative cloud storage auditing.

computing. Users are able to delegate the TPA to perform the verification if their resource-constrained devices could not afford the overload of frequent integrity checks.

Numerous research [17, 18, 19] used game theory [20] to analyze the optimal strategy in cloud storage auditing. Zhou *et al.* [17] proposed a game-theoretic audit model to motivate the CSP to follow the SLA. Based on the Nash equilibrium (NE) of their game model, the CSP has no incentive to cheat if the penalty of being cheating is more than the payoff of being honest. Ismail *et al.* [18] provided a game-theoretic analysis to audit whether the CSP is in compliance with the data replication policy of SLA. According to their analysis results, if the storage cost of one replica become larger, the CSP has less incentive to act honestly. In this case, the TPA will spend more resources on the auditing works. Lin *et al.* [19] proposed a game-theoretic model to analyze the conflict between a user and the CSP, where the user prefers less auditing works and the CSP prefers fewer replications. The evaluation results show that if the user stores the high-value critical data in the CSP, he should choose to audit for $1.0 \leq r \leq 2.1$ in the mild punishment and $1.2 \leq r \leq 2.2$ in the severe punishment, where r is the number of replicas in CSP's replication policy.

Niyato *et al.* [21] presented a hierarchical cooperative game model to improve the efficiency and utilization of the resource in multiple CSPs. In this model, a group of CSPs can cooperate to establish a resource sharing coalition. Mashayekhy *et al.* [22] introduced a cloud federation formation game, where CSPs cooperatively offer cloud IaaS services to fulfill users' requests on virtual machine instances. CSPs can decide to form coalitions to dynamically allocate their resources, *e.g.*, cores, memories, and storages.

3. SYSTEM MODEL

Overview. Our model consists of three entities, as illustrated in Fig. 1: a set of users $N = \{1, 2, ..., n\}$, a cloud storage provider $\mathcal{CSP}$, and a social networking service $\mathcal{SNS}$. Users are individual consumers and organizations, who store their data in $\mathcal{CSP}$. $\mathcal{CSP}$ is a service provider of data storage, who offers data access interfaces to let users store and retrieve their data. Furthermore, $\mathcal{CSP}$ allows users remotely verify the integrity of their data by using the auditing schemes. $\mathcal{SNS}$ is a computing platform, where users build their friend relationships and upload their auditing results, *e.g.*, Facebook, Twitter, and Google+.

In $\mathcal{SNS}$, the friend relationship between two users happens after one accepts another's friend establishment request. The friend relationship is symmetric if it exists. That is, for users $i, j \in N$, if i is j's friend, it implies that j is i's friend. According to users' friend relationships, a friend graph is constructed in $\mathcal{SNS}$. In the friend graph, two users are connected if they are friends. We regard the friend graph as an undirected graph because

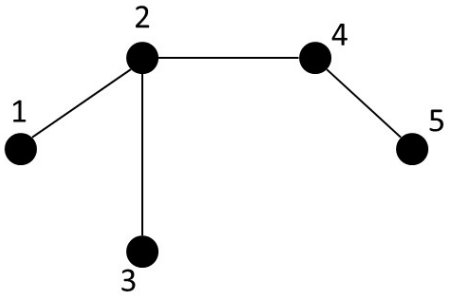

Fig. 2. The example of friend graph.

of the symmetric property.

For a user, in order to evaluate his opinion about $\mathcal{CSP}$, a *CSP score* is calculated in $\mathcal{SNS}$. A user's CSP score depends on his auditing results. The user's CSP score will increase after he uploads a pass result to $\mathcal{SNS}$. On the contrary, his CSP score gets lower if he gets a fail result in his auditing.

Furthermore, $\mathcal{SNS}$ constructs a trust graph based on the friend graph. In the trust graph, for every two friends, a trust value is assigned between them according to the difference of their CSP scores. In general, if two friends' previous opinions about an item are similar, they can build a high trust for the same item. For example, if formerly two friends have similar opinions about the food of a restaurant, one can highly trust another's opinion before he goes to the same restaurant. In the trust graph, the trust value between two friends is high if their CSP scores are close. When the trust value between users is strong enough, one can refer to another's opinion. Notice that we do not consider the friend-of-friend relationship. In the trust graph, every user does not set any trust value between his friend's friend.

Notation. Here, we establish notations to denote the components in $\mathcal{SNS}$. Let $v_i(t)$ denote user i's auditing result at time t. If $\mathcal{CSP}$ passes i's auditing, $v_i(t) = true$; otherwise, $v_i(t) = false$.

We denote $r_i(t)$ as user i's CSP score at time t. For user $i \in N$, $r_i(t)$ is evaluated by the following equation.

$$r_i(t) = \begin{cases} r_i(t-1) + \alpha & \text{if } v_i(t) = true \\ r_i(t-1) - \alpha & \text{if } v_i(t) = false \end{cases} \tag{1}$$

where $r_i(0) = 0$ and α is the reputation parameter.

For example, for $t = 5$ and $\alpha = 0.5$, assume that user i's auditing results are $v_i(1) = true, v_i(2) = true, v_i(3) = true, v_i(4) = false$, and $v_i(5) = true$. According to Eq.(1), we have $r_i(1) = r_i(0) + 0.5$, $r_i(2) = r_i(1) + 0.5$, $r_i(3) = r_i(2) + 0.5$, $r_i(4) = r_i(3) - 0.5$, and $r_i(5) = r_i(4) + 0.5$. As a result, we get $r_i(5) = 1.5$.

Let $\mathcal{FG} =< V, E >$ be the friend graph, where V is the set of users and edges $e_{ij} \in E$ if users i and j are friends. For example, for a user set $N = \{1, 2, 3, 4, 5\}$, users 1 and 2 are friends. Users 1, 3, 4 are user 2's friends. Users 4 and 5 are friends. The friend graph of this example is shown in Fig. 2.

Let $\mathcal{TG} =< V, E, w >$ be the trust graph, where V and E are the same as those defined in $\mathcal{FG}$. For two friends i and j, the weight function w assigns the opinion distance between them. The weight of edge e_{ij} is defined as $w(e_{ij}) = u(|r_i(t) - r_j(t)|)$, where u is one of the following three functions u_1, u_2, and u_3. Intuitively, for two friends i and

j, if the difference of their CSP scores $r_i(t)$ and $r_j(t)$ is small, $w(e_{ij})$ is close to one. Otherwise, $w(e_{ij})$ is close to zero.

Definition 1 *u_1 is the step function, which is defined as follows.*

$$u_1(x) = \begin{cases} 1 & \text{if } x <= \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where θ is the threshold.

Definition 2 *u_2 is the logistic function, which is defined as follows.*

$$u_2(x) = \frac{1}{1 + e^{\gamma * (x - \theta)}} \tag{3}$$

where γ is the coefficient of growth and θ is the threshold.

Definition 3 *u_3 is the Gaussian function, which is defined as follows.*

$$u_3(x) = e^{-\frac{x^2}{2 * \sigma^2}} \tag{4}$$

where σ is the standard deviation.

For example, we consider the trust graph $\mathcal{TG} =< V, E, w >$, where V and E are based on the friend graph in Fig. 2 and w is the step function u_1 with $\theta = 1$. Assume that we have $r_1(t) = 5, r_2(t) = 4, r_3(t) = 4, r_4(t) = 3$, and $r_5(t) = 1$ at time t. In this case, we get $w(e_{12}) = 1, w(e_{23}) = 1, w(e_{24}) = 1$, and $w(e_{45}) = 0$.

4. AUDITING COALITION GAME

4.1 Game Model Description

Let $K \subseteq N$ be an auditing coalition of users, where $K \neq \emptyset$. The coalition structure $\mathcal{CS} = \{K_1, ..., K_m\}$ is the set of auditing coalitions, such that $\forall i \in N$, i is a member in exactly one of $K_1, ..., K_m$. For example, for user set $N = \{1, 2, 3\}$, there are seven possible auditing coalitions: $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$ and five possible coalition structures: $\{\{1\}, \{2\}, \{3\}\}, \{\{1, 2\}, \{3\}\}, \{\{1, 3\}, \{2\}\}, \{\{1\}, \{2, 3\}\}, \{\{1, 2, 3\}\}$.

Let ϵ be the opinion adoption threshold. For users $i, j \in \mathcal{TG}$, if the weight between them is higher than ϵ, i can trust j's auditing results, and vice versa. We denote $degree(i)$ as the number of i's adjacent vertices with $w(e_{ij}) \geq \epsilon$. In an auditing coalition K, if a user i satisfies $degree(i) = |K| - 1$, he is elected as an opinion leader. The opinion leader is qualified to audit $\mathcal{CSP}$ because his auditing results can be trusted by the other members. According to the leader' auditing results, the others can judge the service quality of $\mathcal{CSP}$ without auditing by themselves.

For an auditing coalition $K \subseteq N$, if one of the users is elected as the opinion leader, the auditing cost of K is estimated to be one unit because only the leader needs to perform the auditing work. Otherwise, if we can not find any opinion leader, the auditing cost of K is estimated to be $|K|$ units because the users of K have to individually audit by themselves. Formally, we define the auditing cost of an auditing coalition as follows.

Definition 4 *For $K \subseteq N$, $c(K)$ is the auditing cost of K, where*

$$c(K) = \begin{cases} 1 & \textit{if an opinion leader is found in } K \\ |K| & \textit{otherwise} \end{cases}$$

For example, given a trust graph in the example of section 3 and set $\epsilon = 0.5$, we get $c(\{1,2,3,4\}) = 1$ since user 2 can be the opinion leader in the auditing coalition $\{1,2,3,4\}$. However, $c(\{2,4,5\}) = 3$ since we can not find any opinion leader in the auditing coalition $\{2,4,5\}$.

Furthermore, for a coalition structure $\mathcal{CS} = \{K_1, ..., K_m\}$, the overall auditing cost is the summation of the auditing cost over $K_1, ..., K_m$. We define the overall auditing cost $C(\mathcal{CS})$ as follows.

Definition 5 *$C(\mathcal{CS})$ is the overall auditing cost for a coalition structure $\mathcal{CS}$, where*

$$C(\mathcal{CS}) = \sum_{K \in \mathcal{CS}} c(K)$$

For example, for the coalition structure $\{\{1,2,3,4\},\{5\}\}$, $C(\{\{1,2,3,4\},\{5\}\}) = c(\{1,2,3,4\}) + c(\{5\}) = 2$.

In the game model, users' entire benefit is the reduction of the overall auditing cost after joining the auditing coalitions. Our objective is to find out a coalition structure with minimal overall auditing cost, which we call an optimal coalition structure $\mathcal{CS}^*$. We define the optimal coalition structure $\mathcal{CS}^*$ as follows.

Definition 6 *An optimal coalition structure $\mathcal{CS}^*$ is a coalition structure, such that*

$$\forall \mathcal{CS}, C(\mathcal{CS}^*) \leq C(\mathcal{CS})$$

In summary, the problem of finding optimal coalition structure is that given the user set N, the objective is to determine a coalition structure $\mathcal{CS}^* = \{K_1, ..., K_m\}$, such that (a) $\forall i \in N$, i exists in exactly one coalition of $K_1, ..., K_m$ and (b) the overall auditing cost of $\mathcal{CS}^*$ is minimal over possible coalition structures.

4.2 One-Iteratioin Algorithm

Algorithm 1 proposes an optimal coalition structure $\mathcal{CS}^*$ in one iteration. The algorithm is executed in $\mathcal{SNS}$ at time $t = \pi$. We assume that the friend graph $\mathcal{FG}$ has been built, and for user $i \in N$, i's CSP score $r_i(\pi)$ has been calculated by using his previous auditing results $v_i(1), v_i(2)$,...,$v_i(\pi)$.

At the beginning of Algorithm 1, for each friend edge in E, the algorithm assigns a weight according to users' CSP scores by using Eqs. (2)-(4) (line 1). After that, the algorithm constructs an initial trust graph $\mathcal{TG}$ (line 2).

In the next phase, the algorithm solves auditing cost minimization problem $ACMP$ to determine a set of auditing coalitions with the minimal auditing cost (line 3). The outputs of $ACMP$ are a coalition set $\mathcal{P}$ and an auditor set $\mathcal{A}$. Next, Reorganization() procedure is called to reorganize $\mathcal{P}$ such that every user in N exists in only one coalition (line 4). Finally, the algorithm assigns $\mathcal{P}$ as the optimal coalition structure $\mathcal{CS}^*$ (line 5).

Algorithm 1 One-iteration algorithm

Input: The set of users N; the friend graph $\mathcal{FG}$; users' CSP scores $r_1(\pi), r_2(\pi), ..., r_{|N|}(\pi)$

Output: The optimal coalition structure $\mathcal{CS}^*$ and the auditor set $\mathcal{A}$;

// Phase I - Initialize the trust graph

1: For $e_{ij} \in E$, $w(e_{ij}) \leftarrow u(|r_i(\pi) - r_j(\pi)|)$;

2: Construct an initial trust graph $\mathcal{TG}$;

// Phase II - Determine the optimal coalition structure

3: $\mathcal{P}$, $\mathcal{A} \leftarrow$ solve $ACMP(\mathcal{TG})$;

4: Reorganization();

5: $\mathcal{CS}^* \leftarrow \mathcal{P}$;

4.2.1 Auditing cost minization problem

$ACMP$ is relaxed from the problem of finding optimal coalition structure. In $ACMP$, we allow that $\forall i \in N$, i exists in more than one auditing coalition. In our game model, let K_i be the set that contains user i and i's adjacent vertices with $w(e_{ij}) \geq \epsilon$. For user $i \in N$, K_i will be taken a candidate when we solve $ACMP$.

The objective of $ACMP$ is to pick auditing coalitions from candidates $K_1, K_2, ..., K_n$, such that the summation of auditing cost is minimal. Let X_i be the indicator variable for K_i, where $X_i = 1$ if K_i is picked; otherwise, $X_i = 0$. Furthermore, in the picked coalitions, we would like to check whether users of N are completely covered. For user $j \in N$, let p_{ij} denote whether the user j exists in the picked coalition K_i, where $p_{ij} = 1$ and $p_{ij} = 0$ for the true case and the false case, respectively. For user $j \in N$, if j exists in the picked coalition K_i, $p_{ij}X_i = 1$. Otherwise, $p_{ij}X_i = 0$. Thus, if user j exists in one or more picked coalitions, $p_{1j}X_1 + p_{2j}X_2 + ... + p_{nj}X_n \geq 1$. Otherwise, if j does not exist in any picked coalition, $p_{1j}X_1 + p_{2j}X_2 + ... + p_{nj}X_n = 0$. We transfer $ACMP$ to the following 0-1 integer linear programming problem (0-1 ILPP).

$$\text{Minimize} \sum_{i=1}^{n} c(K_i)X_i \tag{5}$$

subject to

$$\forall j \in N, \sum_{i=1}^{n} p_{ij}X_i \geq 1, \tag{6}$$

and

$$\forall i \in N, X_i \in \{0, 1\} \tag{7}$$

The object function (Eq.(5)) is to pick coalitions from $K_1, K_2, ..., K_n$ to minimize the sum of auditing cost. The constraints (Eq.(6)) guarantee that for $j \in N$, j exists in one or more picked coalitions. The constraints (Eq.(7)) indicate whether K_i is picked.

Finally, $ACMP$ generates a set of coalitions $\mathcal{P} \subseteq \{K_1, K_2, ..., K_n\}$. In an auditing coalition of $\mathcal{P}$, we call the opinion leader an *auditor*. Let a_K be the auditor of the auditing coalition $K \in \mathcal{P}$. If K_i is in $\mathcal{P}$, we assign user i as the auditor a_{K_i} and put him into the auditor set $\mathcal{A}$. For example, given the friend graph in Fig. 2 and users' CSP scores

$r_1(\pi) = 5, r_2(\pi) = 4, r_3(\pi) = 4, r_4(\pi) = 3, r_5(\pi) = 3$, $ACMP$ may generate $\mathcal{P} = \{K_2, K_4\}$ and $\mathcal{A} = \{2, 4\}$, where $K_2 = \{1, 2, 3, 4\}$ and $K_4 = \{2, 4, 5\}$.

In the original problem of finding the optimal coalition structure, the search space of exhaustive approach is $O(|N|^{|N|})$ [23]. It grows exponentially with the number of users $|N|$. However, in $ACMP$, we search the optimal solution over $K_1, K_2, ..., K_n$. The search space of $ACMP$ can be downsized to $O(|N|)$.

Notice that the optimum solution of $ACMP$ may not be the coalition structure. For example, for $P = \{\{1, 2, 3, 4\}, \{2, 4, 5\}\}$, users 2 and 4 exist in two coalitions. $\mathcal{P}$ is not the coalition structure if any user exists in more than one coalition.

4.2.2 Reorganizing coalition set

In Reorganization() procedure, we would like to reorganize the coalition set $\mathcal{P}$, such that for user $i \in N$, i exists in exactly one coalition. We consider users' individual auditing cost in this procedure. In an auditing coalition K, the auditing cost $c(K)$ is divided among its members. Let $f(i, K)$ be member i's divided cost in the auditing coalition K. If i is an auditor, we set his divided cost be 0 as an incentive for his auditing task. Otherwise, $f(i, K) = \frac{c(K)}{|K|-1}$.

The detail steps of Reorganization() procedure are presented in Algorithm 2. First, for $i \in N$, it determines a coalition K^* with the minimal divided cost over $\mathcal{P}$ (line 2). Notice that if i is not a member of K, we set $f(i, K) = 1$. Then, except K^*, the algorithm removes i from rest coalitions $K \in \mathcal{P}$ if K owns i (line 3).

Algorithm 2 Reorganization()

1: **for all** $i \in N$ **do**
2: $K^* \leftarrow \text{argmin}_{\forall K \in P} f(i, K)$
3: $\forall K \in \mathcal{P}$, except K^*, remove i from K if $i \in K$;
4: **end for**

After reorganizing, the coalition set $\mathcal{P}$ can be an optimal coalition structure. For example, assume that $ACMP$ generates a coalition set $\mathcal{P} = \{\{1, 2, 3, 4\}, \{2, 4, 5\}\}$. Users 1, 3, and 5 will not be moved because the minimal divided cost happens in their staying coalitions. We would like to move users 2 and 4 such that they do not exist in more than one coalition. For user 2, we have $f(2, \{1, 2, 3, 4\}) = 0$ and $f(2, \{2, 4, 5\}) = \frac{1}{3-1} = \frac{1}{2}$. Thus, for user 2, K^* is $\{1, 2, 3, 4\}$. The coalition $\{2, 4, 5\}$ becomes $\{4, 5\}$ after 2 is removed. Next, for user 4, we have $f(4, \{1, 2, 3, 4\}) = \frac{1}{3}$ and $f(4, \{4, 5\}) = 0$. The coalition $\{1, 2, 3, 4\}$ becomes $\{1, 2, 3\}$ after 4 is removed. Finally, the coalition set $\mathcal{P}$ becomes $\{\{1, 2, 3\}, \{4, 5\}\}$. Because $\{\{1, 2, 3\}, \{4, 5\}\}$ is a coalition structure with the minimal auditing cost, we can take it as an optimal coalition structure.

In one iteration, if an optimal coalition structure $\mathcal{CS}^*$ is determined, for $K \in \mathcal{CS}^*$, any user of K does not have incentive to leave K and join another coalition K'. Here, we describe the reason. Let $\mathcal{CS}'$ be a coalition structure that contains the auditing coalitions after a user i left K and joined K'. According to the definition of $\mathcal{CS}^*$, we have $C(\mathcal{CS}') \geq C(\mathcal{CS}^*)$. If $C(\mathcal{CS}') > C(\mathcal{CS}^*)$, for all users, they will pay more overall auditing cost . On the other hand, if $C(\mathcal{CS}') = C(\mathcal{CS}^*)$, users can not get less overall auditing cost on forming K'. For above two cases, if users' entire benefit is considered, the user i prefers to stay in K rather than to leave because the overall auditing cost does not be reduced. On the other hand, if we consider i's individual interest, he will still not leave from the assigned coalition K because the divided cost is already the lowest.

4.3 Multi-Iteration Algorithm

Algorithm 3 is extended from Algorithm 1 to multi-iterations. The algorithm is performed iteratively to change auditing coalitions. With the iteration going on, users continually inspect the auditing result received from their auditors. If a user finds that he receives wrong auditing results, he may join another auditing coalition in the next iteration. In each iteration, the algorithm may generate a new optimal coalition structure. Let m be the termination parameter. If the optimal coalition structure is not changed in m rounds, the algorithm terminates.

First, Algorithm 3 similarly constructs an initial trust graph $\mathcal{TG}$ (lines 1-2). Next, the algorithm iteratively adjusts the optimal coalition structure and the trust graph (lines 3-9). In each iteration, the algorithm follows the lines 3-5 of Algorithm 1 to determine a new optimal coalition structure according to the current trust graph (line 4-6). After that, the algorithm calls AdjustTrustGraph() procedure to adjust the trust graph $\mathcal{TG}$ (line 7). In the end of each iteration, t is set as $t + 1$ (line 8). The iteration terminates until the optimal coalition structure $\mathcal{CS}^*$ has no change in m rounds (line 9).

Algorithm 3 Multi-iteration algorithm.

Input: The set of users $\mathcal{N}$; the friend graph $\mathcal{FG}$; users' CSP scores $r_1(\pi), r_2(\pi), ..., r_{|N|}(\pi)$

Output: The optimal coalition structure $\mathcal{CS}^*$ and the auditor set $\mathcal{A}$;

// Phase I - Initialize the trust graph

1: For $e_{ij} \in E$, $w(e_{ij}) \leftarrow u(|r_i(\pi) - r_j(\pi)|)$;
2: Construct an initial trust graph $\mathcal{TG}$;

// Phase II - Determine the optimal coalition structure

3: **repeat**
4: $\mathcal{P}, \mathcal{A} \leftarrow$ solve $ACMP(\mathcal{TG})$;
5: $P \leftarrow$ Reorganization();
6: $\mathcal{CS}^* \leftarrow \mathcal{P}$;
7: $\mathcal{TG} \leftarrow$ AdjustTrustGraph();
8: $t \leftarrow t + 1$;
9: **until** $\mathcal{CS}^*$ `has no change in` m `rounds`

4.3.1 Adjusting trust graph

We denote $v_{a_K}(t)$ as the auditor a_K's auditing result at time t. For user $i \in N$, if i receives $v_{a_K}(t) = true$ and finds that there are errors in his data after retrieving them, he may miss the errors in the auditing work. In this case, we call that i receives a *false-negative* auditing result. If i receives $v_{a_K}(t) = false$ and gets correct data, $v_{a_K}(t)$ becomes a false alarm for him. This case is a *false-positive* auditing result. On the other hand, i may have *true-positive* and *true-negative* auditing results if $v_{a_K}(t)$ is correct for him. In summary, as in Table 1, users' auditing results fall into four cases.

Table 1. Cases of users' auditing results.

	retrieving wrong data	retrieving correct data
$v_{a_K}(t) = false$	true-positive	false-positive
$v_{a_K}(t) = true$	false-negative	true-negative

For user $i \in N$, after determining the case of his auditing result, he uploads his feedback to $\mathcal{SNS}$. In $\mathcal{SNS}$, according to i's feedback at time t, $r_i(t)$ is updated by using the following equation.

$$r_i(t) = \begin{cases} r_i(t-1) + \alpha & \text{if } i\text{'s feedback at time } t \text{ is true-negative} \\ r_i(t-1) - \alpha & \text{if } i\text{'s feedback at time } t \text{ is true-positive} \\ r_i(t-1) & \text{if } i\text{'s feedback at time } t \text{ is false-negative} \\ r_i(t-1) - 2\alpha & \text{if } i\text{'s feedback at time } t \text{ is false-positive} \end{cases} \tag{8}$$

Here, we describe the procedure to adjust the trust graph $\mathcal{TG}$, which is presented in Algorithm 4. First, for $i \in N$, the algorithm updates i's CSP score at time t according to Eq.(8) (line 1). Next, for $e_{ij} \in E$, a new weight is assigned (line 2). Finally, the algorithm adjusts $\mathcal{TG}$ by using the new weights (line 3).

Algorithm 4 AdjustTrustGraph().

1: For $i \in N$, update i's CSP score $r_i(t)$ according to Eq.(8);
2: For $e_{ij} \in E$, $w(e_{ij}) \leftarrow u(|r_i(t) - r_j(t)|)$;
3: Adjust the trust graph $\mathcal{TG}$;

5. EXPERIMENTAL RESULT

We set the weight function of trust graph to be a logistic function, where $\gamma = 10$ and $\theta = 4$. Furthermore, we set the opinion adoption threshold by $\epsilon = 0.5$. In the multi-iteration experiment, termination parameter m is set as three.

5.1 Experimental Data Generation

WikiLens dataset is the records of feedback from users. The users' feedback records can be used for evaluating a recommendation system. In WikiLens dataset, 408 users give 26937 ratings for 5650 items.

In our experiment, we use the ratings in WikiLens dataset to simulate users' CSP scores at time $t = \pi$. First, we group users by their rated items. The users who rate the same item will be put into the same group. Next, we pick five user sets with $|N| = 10, 20, 30, 40, 50$. Our simulated user sets are shown in Table 2, where r_{max}, r_{min}, r_{mean}, and r_{sd} are the maximum, minimum, mean, standard deviation of the rating value, respectively. In a user set N, for user $i \in N$, we take the rating value of i as i's CSP score $r_i(\pi)$ at time π.

ER random graph [11] $G(V, p)$ is a graph, where the vertices are the elements of V and the edges are generated by the probability p. For user set $N = N_{10}, N_{20}, ..., N_{50}$, we set $V = N$ and generate *loosely-coupled* and *tightly-coupled friend graphs* with $p = 0.3$ and $p = 0.8$, respectively.

5.2 Result Analysis

We use the auditing cost reduction ratio $ACRR$ to analyze the efficiency of auditing cost reduction at time t. Let $UC_{original}$ be the summation of the original auditing cost, where users individually audit by themselves. $UC_{original}$ increases with the number of

Table 2. Simulated user set.

ID	# of users	r_{max}	r_{min}	r_{mean}	r_{sd}
N_{10}	10	5	-3	2.35	3.7
N_{20}	20	5	-3	2.45	3.27
N_{30}	30	5	-3	2.93	2.86
N_{40}	40	5	-3	3.24	2.53
N_{50}	50	5	-3	3.25	2.4

users. Let $UC_{reduced}$ be the summation of the reduced auditing cost at time t, where users cooperatively audit. $ACRR$ is defined by the following equation.

$$ACRR = \frac{UC_{original} - UC_{reduced}}{UC_{original}} \tag{9}$$

Furthermore, we use the accuracy ratio AR to evaluate the accuracy of overall users' auditing works at time t. AR is the proportion of users who receive correct auditing results at time t. Let TP and TN be the number of users who have true-positive and true-negative auditing works at time t, respectively. AR is determined by using the following equation.

$$AR = \frac{TP + TN}{|N|} \tag{10}$$

5.2.1 One-iteration experiment

In the experiment of one-iteration algorithm, we consider loosely-coupled and tightly-coupled friend graphs for user set $N = N_{10}, N_{20}, ..., N_{50}$. Table 3 shows the number of auditing coalitions for the loosely-coupled and tightly-coupled friend graphs in the one-iteration experiment. The results show that for user set $N = N_{10}, N_{20}, ..., N_{50}$, the number of auditing coalitions in the tightly-coupled friend graph is less than that in the loosely-coupled friend graph. It indicates that users are easier to join the auditing coalition with others if they like to make friends.

Fig. 3 shows $ACRR$ of the one-iteration experiment for user set $N = N_{10}, N_{20}, ..., N_{50}$. In Fig. 3, with the number of users increasing from 10 to 50, $ACRR$ is enhanced from 60% to 90% and from 80% to 96% for the loosely-coupled and tightly-coupled friend graphs, respectively. The result illustrates that more users in our cooperative auditing model, the more auditing cost reduction. Furthermore, in Fig. 3, for user set $N = N_{10}, N_{20}, ..., N_{50}$, $ACRR$ is larger if the friend graph is tightly-coupled. Compared to the loosely-coupled friend graph, because users have more choices of forming the auditing coalition in the tightly-coupled friend graph, they can find out an optimal coalition structure with less overall auditing cost.

Fig. 4 shows AR of the one-iteration experiment for user set $N = N_{10}, N_{20}, ..., N_{50}$. Firstly, with the number of users growing from 10 to 50, AR drops from 100% to 88% and from 100% to 90% for the loosely-coupled and tightly-coupled friend graphs, respectively. The result demonstrates that when the number of users grows, the proportion of receiving correct auditing results becomes less. Secondly, for user set $N = N_{30}, ..., N_{50}$, AR of the tightly-coupled friend graph is higher than that of the loosely-coupled friend graph. In the tightly-coupled friend graph, because users have more friends, the probability of choosing the correct one is higher.

Table 3. The number of auditing coalitions in the one-iteration experiment.

	if $\mathcal{FG}$ is loosely-coupled	if $\mathcal{FG}$ is tightly-coupled
N_{10}	4	2
N_{20}	3	2
N_{30}	3	2
N_{40}	5	2
N_{50}	5	2

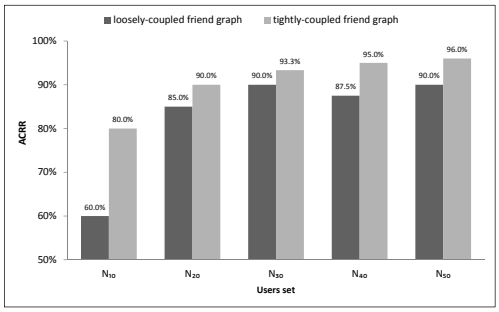

Fig. 3. $ACRR$ of the one-iteration experiment for user set $N = N_{10}, N_{20}, ..., N_{50}$.

5.2.2 Multi-iteration experiment

We consider that multi-iteration algorithm would terminate if $\mathcal{CS}^*$ is not changed in three rounds ($m = 3$). Fig. 5 displays $ACRR$ of the multi-iteration experiment for user set N_{50}. The results show that if the friend graph is loosely-coupled, $ACRR$ drops from 90% to 88% at time $t = 17$. A user forms a new auditing coalition by himself because he can not find any friend whose opinion is close to his. That leads more auditing cost and trims the reduction. On the other hand, in Fig. 5, for the tightly-coupled friend graph, $ACRR$ does not change from start to end of the iteration. If a user has many friends, he could have many auditing coalitions to choose after he left the original one. The overall auditing cost is not changed if the user can join another auditing coalition.

Fig. 6 presents AR of the multi-iteration experiment for the user set N_{50}. The results show that for the loosely-coupled friend graph, AR rises from 88% to 98% at time $t = 13$. It indicates that five users' auditing results become correct after they moved to other auditing coalitions. At time $t = 16$, AR drops to 94%. Totally, there are three users

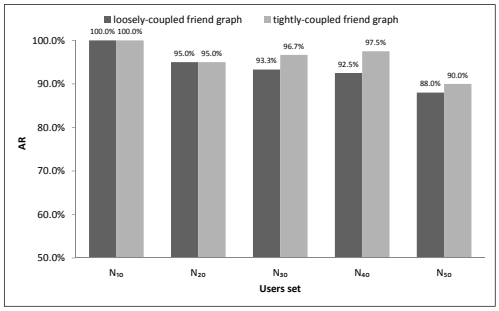

Fig. 4. AR of the one-iteration experiment for user set $N = N_{10}, N_{20}, ..., N_{50}$.

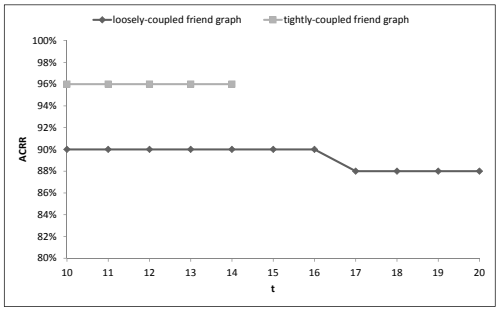

Fig. 5. $ACRR$ of the multi-iteration experiment for user set N_{50}.

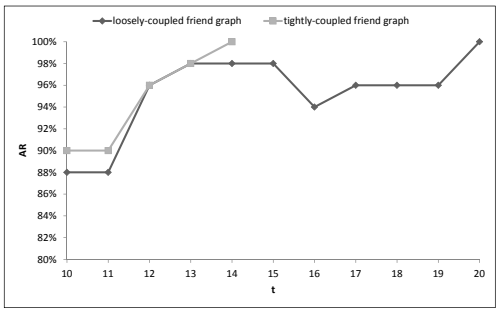

Fig. 6. AR of the multi-iteration experiment for user set N_{50}.

who receive wrong auditing result at that moment. At time $t = 17$, AR increases by 2%. One of the three users receives correct auditing results because he forms a new auditing coalition by himself and becomes an auditor. AR peaks at 100% when the algorithm terminates at time $t = 20$. All users receive correct auditing results at the end of the iteration. On the other hand, in Fig. 6, AR of the tightly-coupled friend graph rises from 90% to 100% at time $t = 14$. It indicates that users spend less time to find correct auditing coalitions if they have more auditing coalitions to choose in each iteration.

Furthermore, the results of Fig. 5 and Fig. 6 indicate that users may lose the reduction in their auditing cost, but they can improve the accuracy of their auditing works in the end of iteration. For example, for the loosely-coupled friend graph, $ACRR$ falls by 2%, but AR grows by 12% in the end of iteration. In this case, one user's extra auditing cost leads to entire auditing results become more precise.

6. CONCLUSION

In this paper, we propose a cooperative model in cloud storage auditing to reduce overall users' auditing cost. In this model, users can utilize their friends' auditing results and avoid paying redundant cost on their auditing works. To motivate users to involve the cooperative auditing model, we design an auditing coalition game by using cooperative game theory. The purpose of auditing coalition game is to assign users into an optimal auditing structure. We present two algorithms to get the optimal auditing structure. In the first algorithm, the optimal auditing structure is determined in one iteration. The second algorithm iteratively adjusts the trust graph. In each iteration, users can change to another auditing coalition to get correct auditing results.

The results of the one-iteration experiment show that if users make more friends,

they can form less auditing coalitions to cover their auditing works. Furthermore, our game model has more reductions in auditing cost with the number of users growing. The maximum of auditing cost reduction peaks at 96% for the user set N_{50} with the tightly-coupled friend graph. In the multi-iteration experiment, some users may bear extra auditing cost, but entire auditing results become more precise. All users can receive correct auditing results in the end of iteration.

According to this work, users can adopt the cooperative model into the existing cloud storage audit mechanisms. For users, if the cost on their auditing works becomes large, they can use one-iteration algorithm to reduce their overall auditing cost. On the other hand, users can apply multi-iteration algorithm to improve the accuracy of auditing results if they store high-value critical data in $\mathcal{CSP}$, $e.g.$, the high-value transaction logs and their family photos. Furthermore, if there is $\mathcal{CSP}'$ in the market, users can omit the process for collecting the auditing history of $\mathcal{CSP}'$. Based on users' existing trust graph, their auditing results for $\mathcal{CSP}$ can be taken as a reference when they play the auditing coalition game for $\mathcal{CSP}'$.

For the future work, we plan to further reduce users' auditing cost by considering the friend-of-friend relationship. Users can utilize the auditing results from their friends' friends. In other words, an auditor could offer more users his auditing results. If two auditors are friends, their auditing coalitions could merge into a big one. Because the number of auditing coalitions becomes less, we expect to get an advanced reduction in auditing cost.

REFERENCES

1. J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Proceedings of International Conference on Intelligent Computing and Cognitive Informatics*, 2010, pp. 380–383.
2. E. Walker, W. Brisken, and J. Romney, "To lease or not to lease from storage clouds," *Computer*, Vol. 43, 2010, pp. 44–50.
3. L. Mastroeni and M. Naldi, "Storage buy-or-lease decisions in cloud computing under price uncertainty," in *Proceedings of the 7th EURO-NGI Conference on Next Generation Internet Networks*, 2011, pp. 1–8.
4. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2007, pp. 598–610.
5. A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Transactions on Information Forensics and Security*, Vol. 10, 2015, pp. 485–497.
6. C. Lin, Z. Shen, Q. Chen, and F. T. Sheldon, "A data integrity verification scheme in mobile cloud computing," *Journal of Network and Computer Applications*, Vol. 77, 2017, pp. 146–151.
7. A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 584–597.
8. J. Li, X. Tan, X. Chen, D. S. Wong, and F. Xhafa, "Opor: Enabling proof of retrievability in cloud computing with resource-constrained devices," *IEEE Transactions on Cloud Computing*, Vol. 3, 2015, pp. 195–205.

9. D. Etheringto, "Amazon aws s3 outage is breaking things for a lot of websites and apps," https://techcrunch.com/2017/02/28/amazon-aws-s3-outage-is-breaking-things-for-a-lot-of-websites-and-apps/, 2017.
10. G. Chalkiadakis, E. Elkind, and M. Wooldridge, "Computational aspects of cooperative game theory," *Synthesis Lectures on Artificial Intelligence and Machine Learn-ing*, Vol. 5, 2011, pp. 1–168.
11. P. Erdős and A. Rényi, "On random graphs i," *Publicationes Mathematicae (Debrecen)*, Vol. 6, 1959, pp. 290–297.
12. Y. Zhu, H. Hu, G. J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, 2012, pp. 2231–2244.
13. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of IEEE INFOCOM*, 2010, pp. 1–9.
14. C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, Vol. 62, 2013, pp. 362–375.
15. C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and R. Kotagiri, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, 2014, pp. 2234–2244.
16. J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end devices in cloud," *IEEE Transactions on Information Forensics and Security*, Vol. 11, 2016, pp. 2572–2583.
17. Z. Zhou, H. Zhang, X. Yu, and J. Guo, "Audit meets game theory: Verifying reliable execution of sla for compute-intensive program in cloud," in *Proceedings of IEEE International Conference on Communications*, 2015, pp. 7456–7461.
18. Z. Ismail, C. Kiennert, J. Leneutre, and L. Chen, "Auditing a cloud provider's compliance with data backup requirements: A game theoretical analysis," *IEEE Transactions on Information Forensics and Security*, Vol. 11, 2016, pp. 1685–1699.
19. C.-Y. Lin and W.-G. Tzeng, "Strategy analysis for cloud storage reliability management based on game theory," *Journal of Computer Security*, Vol. 25, 2017, pp. 1–19.
20. D. Fudenberg and J. Tirole, *Game Theory*, MIT Press, MA, 1991.
21. D. Niyato, A. Vasilakos, and Z. Kun, "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2011, pp. 215–224.
22. L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Transactions on Cloud Computing*, Vol. 3, 2015, pp. 14–27.
23. T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings, "Coalition structure generation: A survey," *Artificial Intelligence*, Vol. 229, 2015, pp. 139–174.

Chung-Yi Lin (林崇頤) received his B.S. degree in Computer Science and Engineering from Yuan Ze University, Taiwan, in 2001; and M.S. degree in Information and Computer Engineering from Chung Yuan Christian University, Taiwan, in 2003. He is currently a Ph.D. student in the Department of Computer Science at National Chiao Tung University, Taiwan, and under the supervision of Prof. Wen-Guey Tzeng. His current research interests include Information Security and Network Security.

Wen-Guey Tzeng (曾文貴) received his B.S. degree in Computer Science and Information Engineering from National Taiwan University, Taiwan, in 1985; and M.S. and Ph.D. degrees in Computer Science from the State University of New York at Stony Brook, USA, in 1987 and 1991, respectively. He joined the Department of Computer Science, National Chiao Tung University, Taiwan, in 1991. His current research interests include Cryptology, Information Security and Network Security.