

A Distributed Trust Evaluation Protocol with Privacy Protection for Intercloud

Yi Dou, Henry C. B. Chan, and Man Ho Au

Abstract—Intercloud seeks to facilitate resource sharing among clouds. To support Intercloud, a trust evaluation framework among clouds and users is required. For trust evaluation, conventional protocols are typically based on a centralized architecture focusing on a one-way relationship. For Intercloud, the environment is highly dynamic and distributed, and relationships can be one-way or two-way (i.e., clouds provide services to each other). This paper presents a distributed trust evaluation protocol with privacy protection for Intercloud. The new contributions and innovative features are summarized below. First, feedback is protected by homomorphic encryption with verifiable secret sharing. Second, to cater to the dynamic nature of Intercloud, trust evaluation can be conducted in a distributed manner and is functional even when some of the parties are offline. Third, to facilitate customized trust evaluation, an innovative mechanism is used to store feedback, such that it can be processed flexibly while protecting feedback privacy. The protocol has been proved based on a formal security model. Simulations have been performed to demonstrate the effectiveness of the protocol. The results show that even when half of the clouds are malicious or offline, by choosing suitable operational parameters the protocol can still support effective trust evaluation with privacy protection.

Index Terms—Intercloud, Trust Evaluation, Privacy, Reputation, Cloud Computing.

1 INTRODUCTION

WITH the rapid advancement of cloud computing, there is an increasing number of cloud services. Each provides different service qualities, pricing and access strategies. Choosing the right cloud services before actually using them is not trivial. In the conventional cloud computing environment, once a cloud user decides to select a cloud service, it is difficult and costly to switch to a new cloud service provider. To address this vendor lock-in problem and to support more cooperative cloud services, Intercloud has been proposed [1], [2], [3], [4]. In the Intercloud paradigm, cloud service providers can process user requests by leveraging services from other clouds [5], [6]. Cloud service providers can share their infrastructure to improve overall resource utilization [7], [8]. Furthermore, applications can be migrated from one cloud service provider to another cloud service provider [9], [10] and workloads can be distributed among clouds for disaster recovery or multi-region application delivery [11], [12]. In this paper, we consider an Intercloud system based on the IEEE P2302 Draft Standard [13], which employs a three-tier architecture, namely, root, exchanges and clouds. The root is a cluster of servers/clouds providing certification and naming services. The clouds provide cloud services to users and to each other. Like Internet exchanges, Intercloud exchanges mediate between the root and clouds. Each cloud should belong to at least one Intercloud exchange. The root, Intercloud exchanges and clouds can communicate with one another through Intercloud gateways by means of Extensible Markup Language (XML)-based messages (e.g.,

based on an Intercloud communication protocol) [14], [15].

The basic Intercloud system can also be extended to support a mobile Intercloud system [16]. In this case, heterogeneous clouds can work collaboratively under a mobile environment so that data, applications and virtual mobile terminals can move across clouds through various handoff processes. In the Intercloud environment, cloud service selection can be made in an ad-hoc, dynamic and distributed manner. For instance, one cloud may want to select a number of reliable clouds to help run a time-consuming program. For mobile Intercloud, a mobile user may want to select a cost-effective cloud service in a foreign city. This makes cloud service selection in an Intercloud environment more challenging. The trustworthiness of cloud services is an important consideration for making cloud selection decision (i.e., knowing the expected performance of a cloud service). Currently, there has been little work done to study distributed trust evaluation for the Intercloud environment. This paper seeks to contribute to this important topic for the development of Intercloud. Trust in a service is generally concerned with a belief in whether the service can be delivered satisfactorily, in accordance with certain trust attributes. In the Intercloud context, a cloud service provider (or user) typically trusts another cloud service provider based on certain trust attributes, such as service reliability, quality of service and service efficiency. Before choosing/using a service, trust evaluation is often conducted based on the feedback of existing users (i.e., reputation-based trust evaluation). Indeed, feedback provided by past cloud users is a good reference for trust evaluation [17], [18]. Based on this feedback or rating, a cloud user can evaluate how likely (e.g., a probability) that a cloud service will be performed as expected. However, the credibility of feedback is often difficult to guarantee [18] as cloud users often avoid leaving honest comments, especially negative ones [19]. The

• The authors are with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.
E-mail: yi.dou@connect.polyu.hk, {cshchan, csallen}@comp.polyu.edu.hk

Manuscript received 29 September 2017; revised 20 May 2018; accepted 07 November 2018.

main reason for this behavior is the unequal status between cloud service providers and cloud users (e.g., a cloud service provider can easily remove negative comments about its services). This problem becomes more serious in the Intercloud environment. As there is more and more mutual co-operation, a cloud user or his/her business could be another type of cloud service provider in future business transactions. This possible mutual relationship makes the privacy requirement even more important in the Intercloud scenario. If feedback information cannot be made private, cloud users may only give positive feedback, as they want to maintain a good relationship or are fearful of retaliation [20], [21]. Hence, it is important to develop an effective and flexible trust evaluation protocol with privacy protection for Intercloud.

Inspired by related work on trust and reputation evaluation, this paper presents a distributed trust evaluation protocol with privacy protection to address the following important requirements. Our contributions are summarized as follows.

- **Cloud user protection.** To encourage honest feedback/ratings and to prevent possible retaliatory attacks, both user identity and user feedback privacy should be protected. Ideally, feedback should not be linked with the user and business privacy of the user (i.e., which user has performed business with which cloud service provider should not be disclosed). Our protocol uses an innovative mechanism to store feedback, and employs homomorphic encryptions [22] and [23] with verifiable secret sharing [24] to protect feedback privacy. Finally, neither the cloud service provider nor the enquirer can obtain individual feedback.
- **Cloud service provider protection.** Malicious users can generate a large volume of misleading feedback or faked ratings to damage the reputation of a cloud service provider. To address this important issue, our proposed protocol allows a cloud service provider to certify a rater's eligibility. Furthermore, as explained later, our protocol allows the filtering of extreme ratings without leaking privacy information.
- **Trust result availability.** Existing distributed protocols typically require all concerned parties to remain online to facilitate feedback collection. This requirement is not practical in the Intercloud environment. The proposed protocol can still function well, even if concerned parties are not available to contribute to trust evaluation.
- **Flexible processing of protected feedback.** To facilitate customized trust evaluation and reduce the influence of misleading ratings, it is desirable to provide a flexible way to subjectively process protected feedback results. For example, suppose there are two sets of ratings: 1, 5, 5, 5 and 4, 4, 4, 4. Although they both give an average rating of 4, one or the other set may be preferred by different enquirers. Our protocol provides an innovative mechanism to store and process ratings in a flexible manner (e.g., assigning a lower weight to de-emphasize or filter extreme ratings) while protecting feedback privacy.

The remaining sections of this paper are organized as follows. Section 2 presents the models and overview. Section

3 explains the feedback computation. Section 4 discusses the trust evaluation protocol. Section 5 presents the security proof and discusses the security analysis. Section 6 presents the simulation settings and Section 7 discusses the simulation results. Section 8 discusses related works. Section 9 concludes the paper and outlines future work.

2 MODELS AND OVERVIEW

In this section, we first discuss the system model, main protocol phases and adversary model with assumptions.

2.1 System Model

Fig. 1 shows the system model or architecture with five main components: cloud service provider (CSP); user/rater; enquirer; distributed feedback storage (FBS) and secret sharing network. Under the Intercloud system model, the CSPs provide cloud services collaboratively to users, and serve each other as well. In general, there are two types of cloud service **users**: consumer users and business users. Consumer users use a cloud service. They have only a one-way trust/service relationship with a CSP (i.e., the service is one-way CSPs serving consumer users). Business users may provide services to each other (i.e., two-way trust/service relationship). After using a cloud service or under certain arrangements, the users can rate the service or a trust attribute (e.g., availability, response time, price, technical support). That means the users are also **raters** during the feedback/rating process. Note that to facilitate the explanation, we focus on evaluating one service or trust attribute. It can easily be extended to evaluate multiple services or trust attributes. For a business user, a human representative of the respective organization can provide the rating/feedback. Furthermore, with advances in intelligent computing technologies, a software agent or software robot can perform the feedback/rating task as well. These software agents/robots can communicate through Intercloud gateways (e.g., using an Intercloud communication protocol with predefined XML-based messages). In this case, the Intercloud system becomes a highly autonomous system. The **enquirers** can be any parties who want to use the trust evaluation protocol to evaluate the trustworthiness of a CSP based on the feedback/ratings.

We assume there is a distributed **FBS** for storing feedback. For the Intercloud system, the FBS can be implemented by the Intercloud exchanges, or by the cloud service providers themselves using a blockchain-based system. Basically, feedback is submitted as a record in the blockchain. With the blockchain consensus mechanisms, feedback integrity can be preserved. Alternatively, other distributed storage systems, with integrity guarantees, can also be used to store feedback. We also assume there is a **secret sharing network** for protecting encrypted feedback. Detailed operations will be explained later. The secret sharing network can be formed by the users/raters to protect their feedback privacy. Alternatively, it can be provided by a third party as a service. In the aforementioned autonomous Intercloud system, secrets can be shared by software agents through the Intercloud gateways. Again, communications can be facilitated by the Intercloud communication protocol. Note that

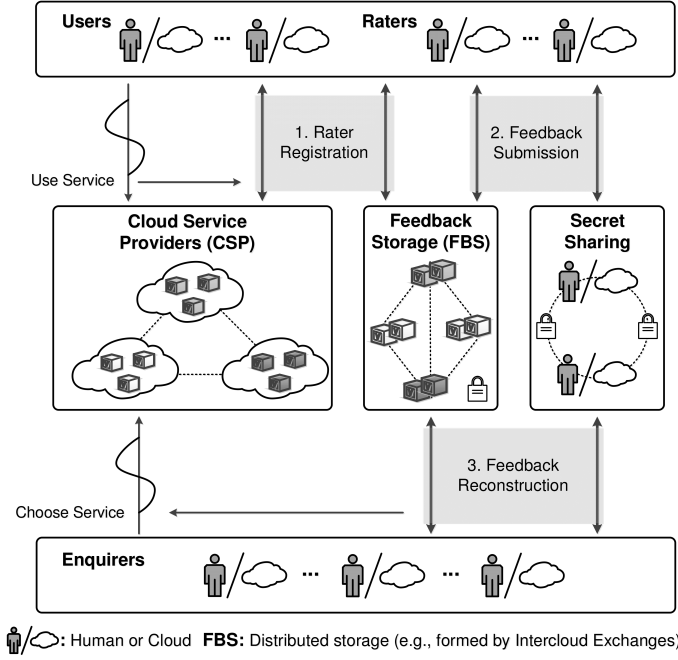


Fig. 1: System architecture and protocol overview.

the secret sharing mechanism is conducted in a distributed manner, without direct interaction between raters. Through a possibly anonymous secret sharing address (SSA) (e.g., a server or Intercloud gateway), each rater only needs to reply to the enquirer by decrypting a certain secret published/provided by the FBS.

2.2 Main Protocol Phases

In this subsection, we give an overview of the three main phases of the protocol. The detailed protocol will be presented in Section 4. As shown in Fig. 1, the first phase is **rater registration**. After using a cloud service provided by a CSP, a (business/consumer/agent) user registers with the FBS as a rater so that feedback/ratings on the service can be submitted (i.e., for a trust attribute). For the registration, it is important to ensure that a rater is a real user. To fulfill this requirement, a user/rater (e.g., the corresponding public key) is certified by the CSP by means of a blind signature [25]. For extra verification, an additional blind signature certification can be provided by a bank or a payment system (i.e., to verify that the user has a payment transaction with the CSP). Note that there can be no linkage between the user identity and rater identity. During registration, the user/rater also needs to provide a secret sharing address (SSA). After rater registration, a secure channel can be set up between the user/rater and the FBS for submitting feedback.

The second phase is **feedback submission**. In this phase, raters choose different secret keys to encrypt their feedback, using a symmetric homomorphic encryption scheme. Due to the additively homomorphic property of the encryption scheme, different raters' encrypted feedback can be added together directly to produce overall feedback, which can only be decrypted with the sum of the raters' secret keys. To support the decryption of the overall feedback while preventing the disclosure of individual feedback, the

raters form a secret sharing network based on a verifiable secret sharing scheme. The encrypted feedback and aforementioned SSAs are maintained by the FBS (i.e., a secure distributed storage to protect data integrity). Detailed approaches will be presented in the Section 4.2. The last phase is the **feedback reconstruction** for trust evaluation. To conduct a trust evaluation based on encrypted feedback, an enquirer progressively sends request to each rater based on SSAs provided/published by the FBS. Note that there can be no linkage between a SSA and the corresponding rater. A rater replies to the request based on the information maintained on the FBS. When there are sufficient replies (i.e., secret shares) from the raters, the enquirer can regenerate the required secret key for decrypting the feedback or rating result. The enquirer can also flexibly process the ratings by using different evaluation weights. Note that the evaluation can be performed while protecting feedback privacy. The detailed protocol will be presented in Section 4.3.

2.3 Adversary Model and Assumptions

In the protocol, we assume that the FBS is reliable for verifying rater identities and ensuring feedback integrity (e.g., by using a blockchain-based system formed by the Intercloud exchanges). That means, once feedback is submitted, it cannot be altered or removed from the FBS. However, the FBS itself may not guarantee privacy protection (i.e., its main purpose is to protect data integrity). We also assume that all communication channels are secure (i.e., communication security is outside our scope). Our security goal is to prevent misbehaving parties from jeopardizing the system operation. As previously mentioned, with the aim of protecting cloud users and CSPs and safeguarding system availability, the scope of this paper is to tackle the following major security attacks using a distributed system.

- **Attack 1: Malicious users.** These malicious users try to affect trust evaluations by submitting misleading feedbacks to the FBS. They may have never used the cloud services before, but pretend to be real users.
- **Attack 2: Selfish raters.** Attacks from selfish raters can be further divided into two types.
 - a) Selfish raters prepare incorrect secret sharing information.
 - b) Selfish raters refuse an enquirer's request for secret key reconstruction, such that the enquirer cannot conduct the trust evaluation.
- **Attack 3: Malicious FBS.** Attacks from a malicious FBS can be further divided into two types.
 - a) It discloses the privacy information of a rater so that the user identity can be found.
 - b) It leaks rater information to a CSP so that negative raters can be identified.
- **Attack 4: Malicious CSPs.** Attacks from malicious CSPs can be further divided into two types.
 - a) It may impersonate a valid enquirer to ask for honest raters to obtain individual feedbacks.
 - b) It may manipulate the FBS and collude with a number of malicious raters to reconstruct the secret keys of honest raters so as to decrypt individual feedback.

3 COMPUTATION OF FEEDBACK RESULTS

Before presenting the trust evaluation protocol in detail, this section first introduces how to compute feedback results for trust evaluation, while protecting feedback privacy.

3.1 Processing of Feedbacks in Plaintext

In this subsection, we first discuss the processing of feedback or ratings in plaintext. Let $\mathbb{R} = \{r_1, \dots, r_n\}$ be the set of users/raters that have used a cloud service. These raters can provide ratings for trust evaluation. As mentioned above, to facilitate the explanation, we focus on one service or trust attribute. There is a set of rating choices $\mathbb{B} = \{b_1, \dots, b_z\}$ for the raters to choose from. Each rating choice represents a different level of satisfaction with the cloud service or a trust attribute (e.g., in a five-star rating system, one-star (\star) indicates not satisfactory and five-stars ($\star\star\star\star\star$) indicates very satisfactory. We define f_l as the feedback of a rater r_l ($1 \leq l \leq n$) for a service or trust attribute. Each rater chooses one of the rating levels in the feedback, that is $f_l \in \{b_1, \dots, b_z\}$. To facilitate the later computation in our scheme, we present each rating choice as a big integer with $z(\lfloor \log n \rfloor + 1)$ binary bits. Each set of $(\lfloor \log n \rfloor + 1)$ binary bits allows the counting on each rating choice. For example, consider that there are four raters ($n = 4$) giving feedback to a five-star rating system. For each rating choice, three binary bits are used for counting purposes (i.e., how many raters chose the choice). That means that each rater's feedback f_l is one of the following ratings:

b_5	$\star\star\star\star\star$	001,000,000,000,000
b_4	$\star\star\star\star$	000,001,000,000,000
b_3	$\star\star\star$	000,000,001,000,000
b_2	$\star\star$	000,000,000,001,000
b_1	\star	000,000,000,000,001

By storing the ratings using the above mechanism, the number of raters for each choice can be found by adding the feedback. For example, raters r_1, r_3, r_4 rate for $\star\star$ and r_2 rates for $\star\star\star\star\star$. By adding their feedback together, we can obtain the following result in binary.

$u_1, f_1 = b_2$	000,000,000,001,000
$u_2, f_2 = b_5$	001,000,000,000,000
$u_3, f_3 = b_2$	000,000,000,001,000
$u_4, f_4 = b_2$	000,000,000,001,000
$R_{sum} = \sum_{l=1}^{l=4} f_l$	001,000,000,011,000
$001_2 = 1_{10}, 000_2 = 0_{10} \quad 011_2 = 3_{10}$	

By converting the binary sum to the corresponding decimal number, we can obtain the number of raters for each rating choice, that is $|b_j|$. In the above example, the binary bits 001 indicate that one rater rated $\star\star\star\star\star$ (i.e., $|b_5| = 1$). Also, the binary bits 011 indicate that three raters rated $\star\star$ (i.e., $|b_2| = 3$). An enquirer can determine the number of raters for each choice by computing the following sum:

$$\mathcal{R}_{sum} = \sum_{l=1}^{l=n} f_l. \quad (1)$$

$$|b_1|, \dots, |b_z| = \text{Convert}(\mathcal{R}_{sum}).$$

\mathcal{R}_{sum} is the feedback sum of raters in the set \mathbb{R} . After converting the binary bits in \mathcal{R}_{sum} to the corresponding

decimal number, the number of raters for each rating choice can be found, that is $|b_1|, \dots, |b_z|$. In subsequent sections, we describe how to obtain \mathcal{R}_{sum} without disclosing individual rating of raters. The trust evaluation result can then be computed as follows:

$$\mathcal{T}r = \frac{\sum_{j=1}^{j=z} w_j \times |b_j| \times j}{n}. \quad (2)$$

$\mathcal{T}r$ represents the trust evaluation result for an enquirer based on the feedback ratings. It is a weighted mean of different rating choices. w_j is the evaluation weight assigned by each enquirer to the j th rating choice. $|b_j|$ is the number of raters choosing the j -th rating choice. Note that besides Equation (2), other similar formulas can also be used. Depending on the enquirer's preference, different evaluation weights can be used for evaluation purposes. Note that if the evaluation weight for all ratings is equal to 1, the result will give the average rating. By using the aforementioned approach, an enquirer can process the ratings more flexibly. For example, suppose that an enquirer wants to choose either cloud 1 or cloud 2. The ratings for cloud 1 and cloud 2 are $\{3, 3, 3, 3\}$ and $\{1, 1, 5, 5\}$, respectively. That means, the average ratings are both 3. However, if the enquirer prefers to choose a cloud service with more five-star ratings, a higher weight can be assigned (e.g., 1 for five-star ratings, 0.8 for other ratings). Thus, the trust evaluation result for cloud 1 is $(0.8 \times 4 \times 3)/4 = 2.4$. And the trust evaluation result for cloud 2 will become $(0.8 \times 2 \times 1 + 1 \times 2 \times 5)/4 = 2.9$. So cloud 2 will be chosen. On the contrary, if an enquirer wants to assign a lower weight to filter or de-emphasize extreme ratings (e.g., 0.8 for one-star and five-star ratings, 1 for other ratings). Then the trust evaluation result of cloud 1 is $(1 \times 4 \times 3)/4 = 3$. And the trust evaluation result of cloud 2 will become $(0.8 \times 2 \times 1 + 0.8 \times 2 \times 5)/4 = 2.4$. Hence, cloud 1 will be chosen. Note that as explained later, this flexible processing of ratings can be performed while protecting feedback privacy.

3.2 Homomorphic Encryption of Feedback

In the last subsection, we have introduced how to process feedback in plaintext. To encourage frank feedback and prevent a retaliatory attack, it is important to protect feedback privacy. In this subsection, we present how to process encrypted feedback for trust evaluation. The majority of the homomorphic encryption schemes are based on asymmetric key encryption. That is, plain text/values are encrypted using the same public key, such that they can be directly added or multiplied together. However, in our scenario, it is difficult for raters to share the same pair of public and private keys. On the contrary, we should allow each rater to use a different secret key to protect feedback privacy. In our protocol, we adopt the symmetric homomorphic encryption scheme proposed in [22]. The encryption of a private feedback f is defined as:

$$c = E_k(f) = [(f + k) \times MK] \bmod \alpha \quad (3)$$

where parameter α is a prime, MK is a master key, k is a random secret key. The decryption of $E_k(f)$ is defined as:

$$f = D_k(c) = [c \times MK^{-1} - k] \bmod \alpha \quad (4)$$

TABLE 1: Notations used in the protocol

Notation	Description
CSP	Cloud service provider
FBS	Feedback storage
u_l	The l th user of CSP
r_l	The l th rater who gives feedback to CSP
f_l	The feedback given by rater r_l to CSP
$pubK_l, privK_l$	The public/private keys of rater r_l
k_l	The secret key of rater r_l for feedback encryption
k_{li}	The secret key share prepared by rater r_l for rater r_i
n	The number of raters in a rater set \mathbb{R} of CSP
m	The minimum number of raters for secret key sum reconstruction
$g^{a_{lm-1}}, \dots, g^{k_l}$	The m commitments for verifying the secret key share of rater r_l
$E_{k_l}(f_l)$	The encrypted feedback of rater r_l
$\mathcal{E}_{pubK_l}(k_{li})$	The encrypted secret key share prepared by rater r_l for rater r_i

where parameter MK^{-1} is the multiplicative inverse of MK modulo e . Assuming that c_1 and c_2 are the ciphertexts of feedback f_1 and f_2 under secret keys k_1 and k_2 , respectively. The homomorphic property of the encryption supports direct computation on the ciphertexts as follows:

$$\begin{aligned}
 c_1 + c_2 &= E_{k_1}(f_1) + E_{k_2}(f_2) \\
 &= [(f_1 + f_2) + (k_1 + k_2)] \times MK \mod \alpha \\
 &= E_{k_1 + k_2}(f_1 + f_2).
 \end{aligned} \tag{5}$$

The result can be decrypted using the key MK^{-1} and the corresponding sum of secret keys $k_1 + k_2$ as follows:

$$f_1 + f_2 = D_{k_1 + k_2}(c_1 + c_2). \tag{6}$$

The property of additive homomorphic encryption shown in Equation (6) allows the enquirer to compute the encrypted feedback as if it were computed based on its plaintexts:

$$C_{sum} = \sum_{l=1}^{l=n} E_{k_l}(f_l) = E_{\sum_{l=1}^{l=n} k_l}(\sum_{l=1}^{l=n} f_l). \tag{7}$$

That is $C_{sum} = E_{SK}(\mathcal{R}_{sum})$ where k_l is the secret key of r_l used to encrypt the feedback f_l . C_{sum} is the encrypted sum of all raters' feedback. $SK = \sum_{l=1}^{l=n} k_l$ is the sum of secret keys privately held by each rater. To decrypt C_{sum} and recover the sum of feedback in plaintext \mathcal{R}_{sum} , the enquirer needs to obtain SK .

As indicated in the Equation (7), to obtain SK , the enquirer needs to ask all concerned raters to provide their secret keys. However, this requirement is unrealistic. Moreover, if the secret key of each feedback is disclosed to the enquirer, feedback privacy cannot be protected. Hence, we adopt secret sharing in [26], [27], which allows a rater to share its secret key through n pieces and with only at least m out of n pieces, the secret key can be recovered. We consider that the raters can share their secret keys with one another. When a certain number of raters is available to share the secret keys, the sum of the secret key SK can be determined.

4 TRUST EVALUATION PROTOCOL

In this section, we discuss the trust evaluation protocol in detail, based on the aforementioned system model and

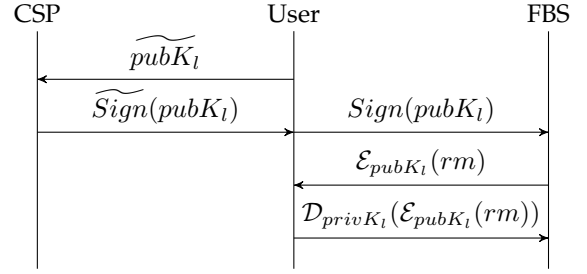


Fig. 2: Process of rater verification.

building blocks. There are three phases in the trust evaluation protocol: rater registration, feedback submission and feedback reconstruction. The notations used in this section to describe the protocol are summarized in Table 1.

4.1 Rater Registration Phase

- 1) $\{\widetilde{Sign}(pubK_l)\} \leftarrow \text{CSP}$. **A user requests for certification by a CSP.** To register with the FBS as a rater, a user $u_l \in \mathbb{U} = \{u_1, \dots, u_n\}$ needs to provide the FBS with proof of identity. The user u_l first generates a pair of public/private keys $\{pubK_l, privK_l\}$ for the certification process. Then, the user u_l asks the CSP to sign the public key $\{pubK_l\}$ by means of a blind signature to prove the right to submit feedback. Basically, the user u_l first sends a "blinded" public key $pubK_l$ to the CSP by combining it with a certain number of random factors. The CSP then returns the blinded signature $\widetilde{Sign}(pubK_l)$ to the user. After unblinding $\widetilde{Sign}(pubK_l)$, the user obtains the signature $Sign(pubK_l)$, which is proof of the rater registration. As previously mentioned, additional certification can be provided by means of a double blind signature (e.g., an additional blind signature by a payment system).
- 2) $\{pubK_l, Sign(pubK_l)\} \leftarrow u_l$. **The user verifies the identity with the FBS.** The user sends the signature $Sign(pubK_l)$ with the public key $pubK_l$ to the FBS for verification. The FBS verifies $Sign(pubK_l)$ with the public key of the CSP. Upon verification, the FBS continues to verify that user u_l has the private key of $pubK_l$ by asking u_l to decrypt a random message rm encrypted with $pubK_l$ (i.e., a challenge and response mechanism). If u_l can correctly decrypt the random message rm , it finally proves that u_l has the right to submit feedback so that the rater registration can be conducted. The verification process is shown in Fig. 2.
- 3) $\{pubK_1, \dots, pubK_n\}, \{SSA_1, \dots, SSA_n\} \leftarrow \text{FBS}$. **The user registers with the FBS as a rater.** Once a user is verified by the FBS, a new rater identity is created for the user. Due to the use of a blind signature (i.e., no linkage between $\widetilde{Sign}(pubK_l)$ and unblinded signature $Sign(pubK_l)$), the user's identity cannot be determined from the rater's identity. The rater also needs to provide a SSA for the secret sharing operation, which has no linkage with the CSP. After user registration, the FBS groups n registered raters together to form $\mathbb{R} = \{r_1, \dots, r_n\}$ and publishes their public keys and SSAs. Note that there can be no linkage between an

TABLE 2: Feedback submission and reconstruction

	Raters	Encrypted Feedbacks	Secret keys ($k_{li} = F_l(i)$)	r_1	...	r_n
Section 4.2	r_1	$E_{k_1}(f_1)$	$F_1(x) = a_{1m-1}x^{m-1} + \dots + k_1$	$\mathcal{E}_{pubK_1}(k_{11})$...	$\mathcal{E}_{pubK_n}(k_{1n})$
	\vdots	\vdots	\vdots	\times	\vdots	\times
	r_l	$E_{k_l}(f_l)$	$F_l(x) = a_{lm-1}x^{m-1} + \dots + k_l$	$\mathcal{E}_{pubK_l}(k_{l1})$...	$\mathcal{E}_{pubK_n}(k_{ln})$
	\vdots	\vdots	\vdots	\times	\vdots	\times
	r_n	$E_{k_n}(f_n)$	$F_n(x) = a_{nm-1}x^{m-1} + \dots + k_n$	$= \prod_{l=1}^{l=n} \mathcal{E}_{pubK_l}(k_{l1})$...	$= \prod_{l=1}^{l=n} \mathcal{E}_{pubK_n}(k_{ln})$
Section 4.3		$+ \downarrow$ $= E_{\sum_{l=1}^{l=n} k_l}(\sum_{l=1}^{l=n} f_l)$	$+ \downarrow$ $= \sum_{l=1}^{l=n} a_{lm-1}x^{m-1} + \dots + \sum_{l=1}^{l=n} k_l$	\downarrow $= \mathcal{E}_{pubK_1}(\sum_{l=1}^{l=n} k_{l1})$...	\downarrow $= \mathcal{E}_{pubK_n}(\sum_{l=1}^{l=n} k_{ln})$
		Dec \downarrow $\mathcal{R}_{sum} = \sum_{l=1}^{l=n} f_l$	Verify \downarrow $g^{\sum_{l=1}^{l=n} a_{lm-1}}, \dots, g^{\sum_{l=1}^{l=n} k_l}$ $\Leftarrow \sum_{l=1}^{l=n} k_l \Leftarrow m \text{ out of } n$	$\mathcal{D}_{privK_1}() \downarrow$ $\sum_{l=1}^{l=n} k_{l1} = \sum_{l=1}^{l=n} F_l(1)$...	$\mathcal{D}_{privK_n}() \downarrow$ $\sum_{l=1}^{l=n} k_{ln} = \sum_{l=1}^{l=n} F_l(n)$

SSA and the corresponding rater, and the SSAs can be published in a group.

4.2 Feedback Submission Phase

In this phase, each rater generates a different secret key for the encryption of its feedback, and shares this secret key with all raters in the same random group/set \mathbb{R} using the (m, n) Shamir's secret sharing method [26]. The secret key share process does not rely on the existing trust relationship between users, since their identities have been anonymized. When certain raters leak the privacy of honest raters, their own secret keys would leak at the same time. Thus, the privacy of raters' feedback is preserved. When a rater wants to update its feedback, it only uses the same secret key to encrypt its new feedback, without updating its secret key shares. Due to the homomorphic property of Shamir's secret sharing, given i th share of each secret key $F_1(i), \dots, F_n(i)$, one can compute the i th share of the sum of n secret keys $F_1(x) + \dots + F_n(x)$. With the distributed collaboration of m out of n raters, the sum of secrets can be reconstructed, where m is the threshold value.

- 1) $\{k_l, \langle k_{l1}, \dots, k_{ln} \rangle, \langle g^{a_{lm-1}}, \dots, g^{k_l} \rangle\} \leftarrow r_l$. **Each rater prepares secret key shares to protect the privacy of the feedback.** To implement the above method, each rater in \mathbb{R} privately constructs a different polynomial with the same degree of $m - 1$, but different random coefficients, and a constant term, which represents the secret key of the rater. Each rater $r_l \in \mathbb{R}$ constructs a polynomial

$$F_l(x) = a_{lm-1}x^{m-1} + a_{lm-2}x^{m-2} + \dots + a_{l1}x + k_l \pmod{\beta}.$$

Rater r_l produces n points on $F_l(x)$ and one for each rater. That is, $k_{li} = F_l(i)$ is the secret key share prepared by rater r_l for rater r_i , $1 \leq i \leq n$. To allow each rater to verify that its secret key share is correctly prepared, we adopt a verifiable secret sharing scheme [24]. Each rater r_l also generates m commitments to the coefficients of $F_l(x)$, that is $\{g^{a_{lm-1}}, g^{a_{lm-2}}, \dots, g^{a_{l1}}, g^{k_l}\}$. The public parameter g is the generator of a cyclic group. In this group, the discrete logarithm is difficult to compute. Rater r_l submits the m commitments $\{g^{a_{lm-1}}, \dots, g^{k_l}\}$ to the FBS.

- 2) $\{\mathcal{E}_{pubK_1}(k_{l1}), \dots, \mathcal{E}_{pubK_n}(k_{ln})\} \leftarrow r_l$. **A rater prepares the encrypted share of the secret key with the rest of the raters.** To prevent the FBS from directly observing the secret key shares, the rater encrypts each secret key share with the public key of the corresponding rater using Paillier encryption [23]. That is, $\mathcal{E}_{pubK_i}(k_{li})$ is the encrypted secret key share prepared by rater r_l for rater r_i . Rater r_l submits n encrypted shares of its secret key $\{\mathcal{E}_{pubK_1}(k_{l1}), \dots, \mathcal{E}_{pubK_n}(k_{ln})\}$ to the FBS, as illustrated in Table 2.
- 3) $\{\prod_{l=1}^{l=n} \mathcal{E}_{pubK_1}(k_{l1}), \dots, \prod_{l=1}^{l=n} \mathcal{E}_{pubK_n}(k_{ln})\} \leftarrow \text{FBS}$. **The FBS computes the product of all encrypted key shares for each rater.** To facilitate the feedback reconstruction process, the FBS computes the product of Paillier-based encrypted key shares for each rater and publishes the product results, as shown in the gray parts of Table 2. Based on the additive homomorphic property of Paillier encryption, the encrypted key shares for the same rater can be multiplied together, which is equal to the encrypted sum of key shares, that is $\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li}) = \mathcal{E}_{pubK_i}(\sum_{l=1}^{l=n} k_{li})$.
- 4) $\{g^{\sum_{l=1}^{l=n} a_{lm-1}}, \dots, g^{\sum_{l=1}^{l=n} k_l}\} \leftarrow \text{FBS}$. **The FBS computes the product of all polynomial commitments for verification.** To allow any m of the n raters to verify the aggregated secret key shares, (i.e., $\sum_{l=1}^{l=n} k_{li}$ is correctly prepared), the FBS computes the product of all raters' polynomial commitments and publishes the product result i.e.,

$$\sum_{l=1}^{l=n} F_l(x) = \sum_{l=1}^{l=n} a_{lm-1}x^{m-1} + \dots + \sum_{l=1}^{l=n} a_{l1}x + \sum_{l=1}^{l=n} k_l \pmod{\beta}.$$

- 5) $E_{k_l}(f_l) \leftarrow r_l$. **A rater submits the encrypted feedback to the FBS.** To submit feedback, rater r_l encrypts the feedback to obtain $E_{k_l}(f_l)$ based on Equation (3). The rater then uploads the encrypted feedback to the FBS, as shown in the gray part of Table 2. The feedback submission process is shown in Fig. 3.
- 6) $\{E_{k_l}(f'_l), \langle \mathcal{E}_{pubK_1}(k'_{l1}), \dots, \mathcal{E}_{pubK_n}(k'_{ln}) \rangle, \langle g^{d_{lm-1}}, \dots, g^{d_{l1}}, g \rangle\} \leftarrow r_l$. **A rater updates the feedback and secret key shares.** A rater can update the feedback by resubmitting new feedback f'_l encrypted using the same secret key k_l to the FBS, that is, to use $E_{k_l}(f'_l)$ to

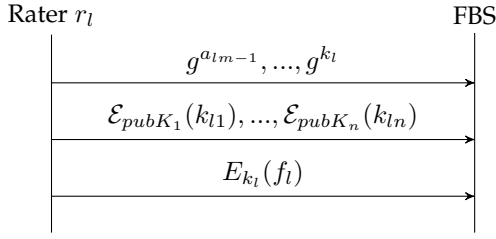


Fig. 3: Process of feedback submission.

replace the old one. To prevent malicious users from colluding to recover the secret keys of honest raters over time, each rater r_l can periodically update the secret key shares by generating a new polynomial $F'_l(x)$ with random coefficients, but a constant term of zero. As a result, an attacker must collude with enough raters within a certain time frame to successfully recover the secret key.

$$F'_l(x) = d_{lm-1}x^{m-1} + \dots + d_{l1}x + 0 \pmod{\beta}.$$

For each remaining rater, a new point on this new polynomial $F'_l(x)$ is computed, that is, $k'_{li} = F'_l(i)$. Each rater then adds the old secret key share with the new point to obtain the new secret key share, that is, $k_{li}^1 = k_{li} + k'_{li} = F_l(i) + F'_l(i)$. Thus, each rater r_l can update secret key shares by submitting n new encrypted secret key shares to the FBS, that is $\{\mathcal{E}_{pubK_1}(k'_{l1}), \dots, \mathcal{E}_{pubK_n}(k'_{ln})\}$. At the same time, rater r_l updates its polynomial commitments by sending $\{g^{d_{lm-1}}, \dots, g^{d_{l1}}, g\}$ to the FBS. Note that the FBS will verify the rater's identity in each submission. After a certain period of time, all submitted encrypted feedback and secret key shares need to be dismissed and can no longer be used. To avoid breaking anonymity and privacy, raters need to generate new secret keys for new feedback, and upload them to FBS.

4.3 Feedback Reconstruction Phase

- 1) $\{E_{k_1}(f_1), \dots, E_{k_n}(f_n)\} \leftarrow \text{FBS}$. **An enquirer gets the encrypted feedback from the FBS.** To evaluate the trustworthiness of a cloud service, an enquirer sends a request to the FBS for the feedback provided by raters. Upon receiving a reply from the FBS, the enquirer computes the encrypted sum of the raters' feedback by adding it together, that is $\mathcal{C}_{sum} = \sum_{l=1}^{l=n} E_{k_l}(f_l) = E_{\sum_{l=1}^{l=n} k_{li}}(\sum_{l=1}^{l=n} f_l)$.
- 2) Request for decryption $\leftarrow \text{Enquirer}$. **The enquirer requests the raters to decrypt the assigned secret key shares.** To decrypt the encrypted sum of feedback, the enquirer needs to reconstruct the sum of the raters' secret keys $\sum_{l=1}^{l=n} k_{li}$. The enquirer progressively sends requests to the raters based on SSAs provided/published by the FBS. In each iteration, the minimum number of raters are chosen and contacted. For example, in the first iteration, m raters are contacted. If only x of them reply, $m - x$ raters will be contacted in the second iteration. Alternatively, the enquirer can broadcast requests

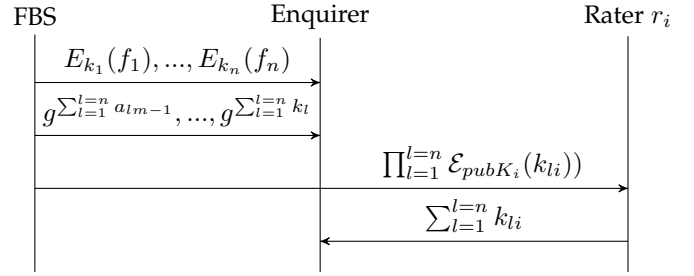


Fig. 4: Process of secret key share collection.

to all raters but some replies may be wasted as only m of them are required.

- 3) $\mathcal{D}_{privK_i}(\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li})) \leftarrow r_i$. **Each rater independently decrypts the product of respective secret shares, and forwards it to the enquirer.** Rater r_i who is willing to contribute to secret key reconstruction gets the product result of the encrypted secret key shares $\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li})$ from the FBS. Due to the additive homomorphic property, $\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li}) = \mathcal{E}_{pubK_i}(\sum_{l=1}^{l=n} k_{li})$. r_i decrypts it using the private key $privK_i$ to obtain $\sum_{l=1}^{l=n} k_{li}$ and sends it to the enquirer. The process of secret key share collection is illustrated in Fig. 4. Note that an honest rater will not decrypt any information other than the encrypted sum of secret key shares assigned to it.
- 4) Secret share verification $\leftarrow \text{Enquirer}$. **The enquirer verifies the secret key shares with the polynomial commitments.** The enquirer obtains the product of all polynomial commitments from the FBS. The enquirer then computes the following value for each rater r_i

$$\begin{aligned} & (g^{\sum_{l=1}^{l=n} a_{lm-1}})^{i^{m-1}} (g^{\sum_{l=1}^{l=n} a_{lm-2}})^{i^{m-2}} \dots (g^{\sum_{l=1}^{l=n} k_{li}})^{i^0} \\ &= g^{\sum_{l=1}^{l=n} a_{lm-1} i^{m-1} + \sum_{l=1}^{l=n} a_{lm-2} i^{m-2} + \dots + \sum_{l=1}^{l=n} k_{li}} \\ &= g^{\sum_{l=1}^{l=n} F_l(i)}. \end{aligned}$$

Accordingly, using the public parameter g and received $\sum_{l=1}^{l=n} k_{li}$, the enquirer computes $g^{\sum_{l=1}^{l=n} k_{li}} \pmod{\beta}$. By comparing $g^{\sum_{l=1}^{l=n} F_l(i)} \pmod{\beta}$ and $g^{\sum_{l=1}^{l=n} k_{li}} \pmod{\beta}$, the enquirer can verify whether $\sum_{l=1}^{l=n} k_{li}$ can correctly reconstruct the sum of all raters' secret keys.

- 5) $\sum_{l=1}^{l=n} k_{li} \leftarrow \text{Enquirer}$. **The enquirer reconstructs the raters' encrypted secret keys.** When there are at least m successful secret share verifications (i.e., from m raters), the sum of n raters' secret keys can be reconstructed using the Lagrange interpolation method, that is, $\sum_{l=1}^{l=n} k_{li}$.
- 6) $\sum_{l=1}^{l=n} f_l \leftarrow \text{Enquirer}$. **The enquirer decrypts the encrypted sum of feedback.** Finally, using the secret keys, the enquirer can decrypt \mathcal{C}_{sum} to obtain $\mathcal{R}_{sum} = \sum_{l=1}^{l=n} f_l$. By converting the respective binary digits of \mathcal{R}_{sum} to the corresponding decimal numbers, the number of raters for each rating choice can be determined. The enquirer finally obtains the trust evaluation result using different evaluation weights: $\mathcal{T}r = (\sum_{j=1}^{j=z} w_j \times |b_j| \times j) / n$.

5 SECURITY EVALUATION

We analyze the security of our proposed Intercloud trust evaluation protocol based on the widely used simulation paradigm [28] in a manner similar to that seen in [29]. In this paradigm, a system is secure if its output distribution approximates an ideal system for all possible input distributions. More specifically, we first define an ideal system where all computations are conducted by a trusted party, \mathcal{T} . Furthermore, all participants communicate through \mathcal{T} via a secure communication channel. As such, this hypothetical ideal system is secure, assuming \mathcal{T} is honest. A real-world system is secure if the output distribution of the system is the same as that of an ideal system for all possible input sequences. A complete security analysis in this paradigm then consists of two parts, namely, a security model and security proof. The former gives the definition of an ideal system, and the latter asserts that the input-output distribution of a real-world system is the same as that of the ideal system.

5.1 Security Model

An Intercloud trust evaluation protocol consists of rater registration, feedback submission and feedback reconstruction phases with relevant parties being the FBS, CSP, users/raters and enquirers. We give the specifications of the ideal-world system as follows.

- **Rater registration** Each user $u \in \mathbb{U} = \{u_1, \dots, u_n\}$ sends a request to register as a rater for a CSP to \mathcal{T} , who forwards the request to the CSP concerned. The response from the CSP is sent back through \mathcal{T} . Upon receiving approval, u sends another request to \mathcal{T} for the registration as a rater with FBS. \mathcal{T} first checks whether the request is legitimate, based on the CSP response. If the request is valid, \mathcal{T} notifies FBS that one eligible user is requesting to give feedback for this particular CSP. Note that \mathcal{T} will not reveal the identity of u to FBS. If FBS accepts this request, \mathcal{T} creates a new rater identity r for u and informs u and FBS. FBS randomly groups raters together to obtain a set of raters $\mathbb{R} = \{r_1, \dots, r_n\}$. The set, \mathbb{R} , is made known to all raters within the set through \mathcal{T} .
- **Feedback Submission** Each rater $r \in \mathbb{R} = \{r_1, \dots, r_n\}$ sends a request to \mathcal{T} for rater set \mathbb{R} that includes r . \mathcal{T} first checks whether the requester is rater r . Once it passes the check, \mathcal{T} returns \mathbb{R} to it. Rater acknowledges set \mathbb{R} to \mathcal{T} . \mathcal{T} informs the FBS that rater r 's \mathbb{R} . After a period of time, each rater r submits its feedback f to \mathcal{T} . Then, after \mathcal{T} confirms the requester is rater r , it will store the feedback and inform FBS that rater r has submitted feedback, but without informing FBS about the content of the rater's rating choice.
- **Feedback Reconstruction** An enquirer asks \mathcal{T} for the CSP feedback given by its past users. \mathcal{T} forwards the enquirer's request to all raters in the \mathbb{R} to ask whether they support the reconstruction of the feedback results. When there is a sufficient number of approval responses from raters, \mathcal{T} calculates the sum of the feedback given by raters in the group \mathbb{R} and returns this sum to the enquirer. Then \mathcal{T} informs FBS that an enquirer has obtained the sum of the feedback in the group \mathbb{R} .

Upon completion of the three phases, all participants output the protocol outcome in each of the above three phases. Obviously, the system in the ideal world is secure, assuming \mathcal{T} follows the specifications completely and that communication between \mathcal{T} and the participants is secure. In particular, CSP and FBS will not be able to obtain individual feedback. CSP also cannot learn the raters' identity. To show that our proposed Intercloud trust evaluation protocol achieves its security goals, it suffices to prove that for all possible execution sequences and all possible inputs, the output distribution of our proposed system is the same as the above ideal system.

Following the above intuition, we define the security of an Intercloud trust evaluation protocol as follows. Let \vec{x} denote the set of inputs for all participants. Let \mathcal{M} denote the set of dishonest participants controlled by adversary \mathcal{A} . To be more specific, we use \mathcal{A}_I and \mathcal{A}_R to denote an adversary in the ideal and the real worlds respectively. Honest participants are denoted as \mathcal{H} . While \mathcal{H} follows the protocol faithfully, \mathcal{A} may act arbitrarily. Let $REAL_{\mathcal{A}_R}(\kappa, \vec{x})$ (*resp.* $IDEAL_{\mathcal{A}_I, \mathcal{S}}(\kappa, \vec{x})$) denote the probability distribution of joint outputs of honest clouds and adversary \mathcal{A}_R (*resp.* adversary \mathcal{A}_I) on inputs \vec{x} and security parameter κ in the real world (*resp.* in the ideal world).

Definition 1. An Intercloud trust evaluation protocol is secure, if for all probabilistic polynomial time adversaries \mathcal{A}_R , there exists a corresponding \mathcal{A}_I in the ideal world such that:

$$\{REAL_{\mathcal{A}_R}(\kappa, \vec{x})\}_{\kappa \in N} \approx \{IDEAL_{\mathcal{A}_I}(\kappa, \vec{x})\}_{\kappa \in N} \quad (8)$$

where \approx represents indistinguishability of two distributions.

5.2 Security Proof

The goal of the security proof is to show that for all \mathcal{A}_R , there exists a corresponding \mathcal{A}_I . To accomplish the goal, we construct a simulator, \mathcal{S} , who plays the role of all honest parties in the view of \mathcal{A}_R , and plays the role of \mathcal{A}_I in the ideal world. That is, \mathcal{S} represents the honest parties to interact with \mathcal{A}_R in the real world, in order to learn about the attacks of \mathcal{A}_R . At the same time, \mathcal{S} represents the dishonest users/raters, CSP, FBS and enquirer to interact with \mathcal{T} in the ideal world. Then, we are going to show that the input-output distribution of the two worlds is indistinguishable. Based on the adversary model analyzed in Section 2.3, in the real world, adversary \mathcal{A}_R might corrupt all participants (i.e., users/raters, CSP, FBS and enquirer) in all three phases of our protocol.

Below we discuss the various attacks presented in our adversary model (Section 2.3) and that if successful, would cause the outputs of the ideal and real world to be distinguishable. Then, we present arguments that these cases can only occur with negligible probability.

- **Rater registration**

- 1) (*Attack 1*) Malicious user in the real world who does not obtain the signature assigned by CSP can register as a rater and submit misleading feedback. In the ideal world, \mathcal{T} will check CSP's approval for a rater registration request, and thus this attack cannot happen. In our

protocol, we assume that FBS is reliable for rater verification and needs to verify the signature with a CSP public key. Consequently, a malicious user only becomes a rater if it can forge a signature of the CSP. We would like to remark that it cannot become a rater by stealing signatures from valid users. The reason is that in our protocol, FBS will validate whether the signature holder is the real user by a challenge-response protocol in which the rater is required to decrypt the ciphertext of a random message. Since a malicious user does not have the private key of the real signature holder, it cannot decrypt the message and cannot pass the check on FBS. As a result, this scenario will not occur.

- 2) (Attack 3a) *Malicious FBS maintains raters' information, which violates business privacy. FBS can help CSP identify raters that gave negative feedback.* In the ideal world, all users' user identity information is kept secret by \mathcal{T} and thus, this attack cannot occur. In our protocol, we adopt a blind signature to prevent this from happening in the real world. Specifically, possession of an unblinded signature only demonstrates that the signature holder is a legitimate user of the CSP, but it cannot be linked to the specific blind signature generation.

• Feedback Submission

- 1) (Attack 2a) *Selfish raters may prepare incorrect secret sharing information and submit it to the FBS.* In the ideal world, \mathcal{T} stores a single feedback value and seeks approval from raters before releasing it. In other words, this attack cannot happen. In our protocol, we adopt verifiable secret sharing to prevent this from happening. Specifically, anyone can verify whether the secret key shares of each rater has been correctly prepared. As such, this attack also cannot happen in the real world.
- 2) (Attack 3b) *Malicious FBS may help CSP to identify raters that gave negative feedbacks.* Malicious CSPs and FBS colludes to obtain actual value of individual feedback. In the ideal world, all feedback is maintained by \mathcal{T} . FBS only knows whether a rater has provided its rating or not, and thus this attack cannot happen. In our protocol, we adopt homomorphic encryption, so that all feedback and secret key shares are encrypted by the secret key and public key of honest raters. Therefore, only the honest raters themselves can decrypt it. The FBS cannot directly deduce the plaintext of these encrypted values and in the FBS's view, these values are indistinguishable from random values. Thus, this attack also cannot happen in the real world.

• Feedback Reconstruction

- 1) (Attack 2b) *Selfish raters refuse the enquirer's request for secret key reconstruction, such that the enquirer cannot reconstruct the feedback sum.* This attack can occur in both the real world and the ideal world, and depends on the number of honest raters versus selfish raters. In the next section, we present an analysis to show feedback reconstructions that fail with very low probability under reasonably chosen parameters.

- 2) (Attack 4a) *Malicious CSP pretends to be an enquirer to ask for honest raters to decrypt the individual secret key share of a specific rater so as to obtain its individual feedback.* In the ideal world, \mathcal{T} checks the identity of the enquirer to prevent this from happening. In our protocol, this will also not happen, since an honest rater will refuse this kind of request from an enquirer.
- 3) (Attack 4b) *Malicious CSP colludes with FBS and n raters.* As long as n is smaller than the threshold, no information is leaked since secret reconstruction is impossible. In the case that n is greater than or equal to the threshold, the real world adversary \mathcal{A}_R can reconstruct the secret key of honest raters and obtain individual feedback f without being noticed. In the ideal world, honest raters will be notified by \mathcal{T} of the fact that someone attempted to reconstruct feedback. In this case, the two world will be distinguishable. We define event fail as the scenario where \mathcal{A}_R obtains individual feedback without being noticed.

In conclusion, the only case in which the outputs of the real and ideal worlds are distinguishable is when event fail occurs in the feedback reconstruction phase. In the next section, we will analyze the probability of a successful attack of this kind, in which the adversary obtains individual feedback from our protocol.

6 SIMULATION SETTINGS

In this section, we conduct simulations to evaluate our protocol, focusing on the security aspects. In particular, the aim is to evaluate how the protocol can resist collusion attack from malicious raters and maintain the availability of trust results with the selfish raters.

Choice of Parameters. We first use a weighted directed graph $G(N, p, q)$ to describe the possible connections between users and CSPs in the Intercloud scenario. A set of vertices N in the graph G represents the set of users/raters and CSPs in the Intercloud. Note that each vertex can be a user as well as a CSP. When a user interacts with a CSP, a directed edge will be added from the cloud vertex to the user vertex on the graph. We define parameter $p \in [0\%, 100\%]$ as the density of graph G , which is the ratio between the actual number of edges and the maximum number of edges in the graph G (i.e., $N(N-1)$). A larger p indicates there are more interactions between users and CSPs. And the weight of each edge in the graph is the feedback given by a rater to a CSP. We define the parameter $q \in [0, 100]$ as the average feedback given by a rater to a CSP, which is also the average weight of an edge in the graph. A larger q shows that the rater tends to give more positive feedback to the CSP. Fig. ?? shows a simple example of the structure of the generated graph G . There are three vertices, each acting as CSP and user at the same time (i.e., $N = 3$). There are four edges.

Thus, the graph density is $p = \frac{4}{6} = 67\%$. Each edge weight is feedback given by a rater to a CSP. In this graph, the average feedback is the average weight of all edges $q = \frac{(100+80+60+40)}{4} = 70$. We assume that all raters adopt our

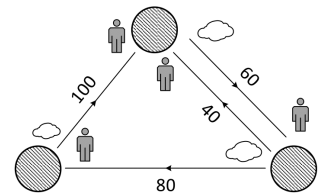


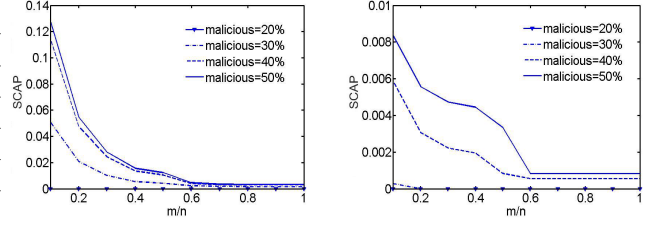
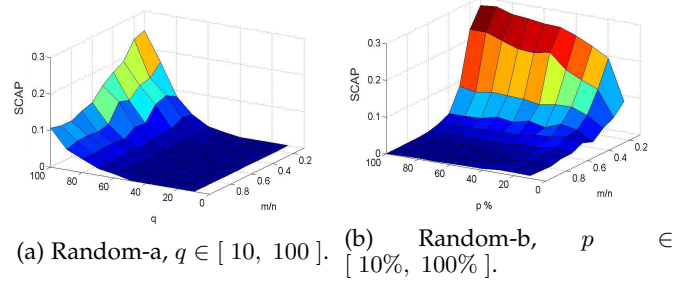
TABLE 3: Graphs characteristics

	Advogato	Robots	Random	
			Random-a	Random-b
Vertices N	5,417	1,732	5,417	1,732
Density p	0.17%	0.12%	0.17%	[10%, 100%]
Avg. feedback q	80.12	70.46	[10, 100]	70.46

protocol to protect the privacy of their feedback. By labeling a portion of raters as malicious or selfish, we evaluate the privacy protection, trust result availability and efficiency of our protocol. Three graphs, denoted by Advogato, Robots, Random, are used in our simulation. Their characteristics are summarized in Table 3.

Trust graphs. There is more mutual co-operation in the Intercloud environment. However, current trust feedback datasets about cloud service only collect feedback given from users to CSPs, which is not a mutual evaluation or two-way trust/service relationship. Currently, no dataset about the two-way Intercloud relationship is available. Thus, we choose to evaluate our protocol based on two trust graphs Advogato and Robots which are commonly adopted in the analysis of any trust-related protocols [19]. These two datasets include a large number of mutual evaluations records, which is close to the two-way trust/service relationship between CSPs and users in the Intercloud environment. For instance, in the Advogato community, each member is a free software developer, and evaluates other developers with different rating levels. Robots community follows the same trust metric. The rating choices are *master*, *journeyer*, *apprentice*, and *observer*, in which *master* is the highest level and *observer* is the lowest or default level for a new account. We downloaded the latest dataset (Jul 07, 2014) of these communities from trustlet.org and used them to build graphs Advogato and Robots respectively, representing the possible relationships between users and CSPs in the Intercloud environment.

Random graph. As shown in Table 3, the graph density and average weight of the two trust graphs Advogato and Robots are similar. To supplement our simulation on the influences of graph density and average feedback, we also use another two graphs (denoted by Random-a and Random-b) with various parameters in our simulations. These two graphs are generated by randomly connecting between vertices (i.e., CSPs and users) and randomly assigning different weights (i.e., feedback) on edges. It is reasonable that several large CSPs would serve as "hubs" of the Intercloud. They have many more users than the rest of the CSPs. Thus, we model the Intercloud as a scale-free network in the random graph. We generate a random variable with Pareto distribution, which represents the outdegree of the vertex (i.e., the number of users interacting with the same CSP). For each outgoing edge, we randomly pick a vertex as its destination. We consider that the majority of raters would choose the middle rating levels (e.g., *apprentice* and *journeyer*). And a small number of raters choose very low or very high rating levels (e.g., *observer* and *master*) as their feedback to the CSPs. Thus, the weights of edges in the random graph (i.e., feedback given by raters to CSPs) are assigned with a random variable following a normal distribution.

(a) Advogato, $q = 80.12$.(b) Robots, $p = 0.12\%$.Fig. 5: Effect of increasing m/n ratio on the successful collusion attack probability (SCAP) in trust graphs.(a) Random-a, $q \in [10, 100]$. (b) Random-b, $p \in [10\%, 100\%]$.Fig. 6: Effect of increasing m/n ratio on the successful collusion attack probability (SCAP) in random graphs.

Random-a is a supplement to Advogato to evaluate protocol performance under various average feedback. Random-b supplements Robots to analyze the influence of density conditions. For Advogato and Robots, we build the graphs from the dataset after a pre-processing step that converts the rating level into the weight of the edges, by the following rule: *master* level = 100, *journeyer* level = 80, *apprentice* level = 60 and *observer* level = 40. Then each cloud has a different size of n users, m of which is the threshold number for feedback reconstruction. In subsequent simulations, we vary m/n ratio $\in [0.1, 1]$, with an increment of 0.1.

7 SIMULATION RESULTS

7.1 Protection against Collusion Attack

In this subsection, we evaluate the relationship between m/n and successful collusion attack probability, under different percentages of malicious raters. We calculate successful collusion attack probability as the percentage of honest raters whose feedback is leaked during the attack. A collusion attack is launched by malicious raters. They intentionally leak the secret key share information of honest raters. When a malicious CSP colludes with at least the threshold number of raters, it can illegally reconstruct the secret key to decrypt the individual feedback of honest raters. In the security analysis in Section 5.2, we have defined this as an event failure, and if this happens, the real world and the ideal world are distinguishable. Thus, the first simulation seeks to analyze the probability of this event failure in a practical setting. We use successful collusion attack probability to represent this event failure probability.

Raters in the network with a low reputation score are considered to be malicious. They have a higher possibility of

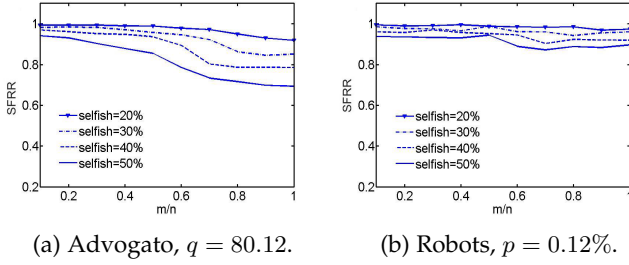


Fig. 7: Effect of increasing m/n ratio on successful feedback recovery rate (SFRR) in trust graphs.

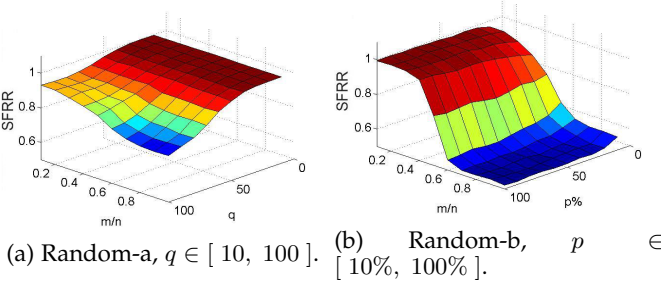


Fig. 8: Effect of increasing m/n ratio on successful feedback recovery rate (SFRR) in random graphs.

colluding with each other to recover the feedback of honest raters. For the trust graphs, we set the malicious raters with reputations ranked in the lowest $\{20\%, 30\%, 40\%, 50\%\}$ of the entire network. To test the worst case scenario, we set 50% of raters as malicious in the random graphs. In Random-a, we fix $p = 0.17\%$ (same density of Advogato) and change q from 10 to 100, with an increment of 10. We use $q = 70.46$ for Random-b (same average feedback of Robots) and vary its p from 10% to 100%, with an increment of 10%.

The results for the trust and random graphs are shown in Fig. 5 and Fig. 6, respectively. It can be seen that even if the network contains 50% malicious raters, the successful collusion attack probability (i.e., the probability of event failure) will be less than 0.1 as long as $m/n > 20\%$ and $q < 90$ (i.e., the average feedback values in the entire network). In the design of our protocol, each rater relies on other raters in the same set to share its secret key. When some of the raters collude to recover other raters' secret keys, their own secret keys are also disclosed. We consider that the majority of raters seek to protect their privacy. In addition, to recover raters' private feedback, CSPs need to inject a high enough number of malicious raters. Since these are real companies, many should have good corporate governance. The real-world collusion attack probability should be lower than the above results. Hence, the proposed protocol should be effective in handling collusion attacks.

7.2 Feedback Recovery Rate of Our Protocol

In this subsection, we discuss the relationship between m/n and the successful feedback recovery rate, under different percentages of selfish raters. The decryption of the secret key sum relies on the support of raters. This

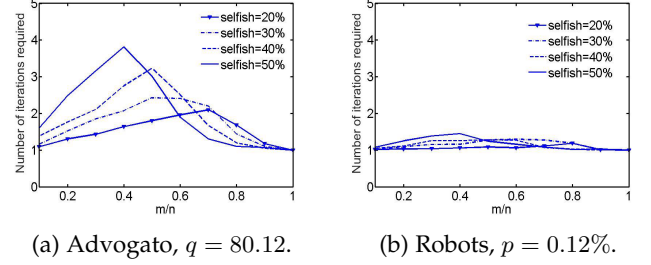


Fig. 9: Effect of increasing m/n ratio on the average number of iterations required to obtain the secret key shares during the query phase in trust graphs.

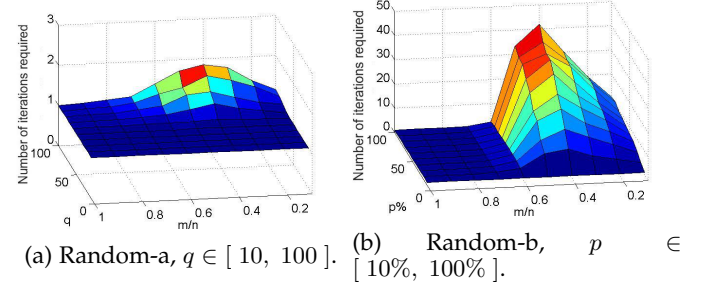


Fig. 10: Effect of increasing m/n ratio on the average number of iterations required to obtain the secret key shares during the query phase in random graphs.

simulation seeks to study whether the protocol can still accurately compute the trust result when there are selfish raters who are not willing to participate in the secret key sum reconstruction. Let $|\mathcal{T}r|$ ($w_j = 1$) denote the number of CSPs' trust results reconstructed from all raters in the network. Let $|\mathcal{T}r'|$ ($w_j = 1$) denote the number of CSPs' trust results recovered only from the raters who are not selfish. We define a successful feedback recovery rate as $\frac{|\mathcal{T}r'|}{|\mathcal{T}r|}$. In the following simulations, selfish raters are randomly assigned. For the trust graphs, we run with selfish raters $\in \{20\%, 30\%, 40\%, 50\%\}$. For the random graphs, we consider that 50% of raters in the network are selfish. Likewise, we evaluate the protocol recovery rate under various average feedback and network density by setting $q \in [10, 100]$ and $p \in [10\%, 100\%]$.

The feedback recovery rates of the protocol for the trust and random graphs are illustrated in Fig. 7 and Fig. 8, respectively. It can be seen that the higher the network density p , the higher average feedback q and the greater m/n all lead to a lower feedback recovery rate. Under the same percentage of selfish raters in the network, the network density p has more influence than parameter q in the successful feedback recovery rate, as shown in Fig. 8. In addition, when there are 50% of selfish raters in a network, the successful feedback recovery rate will be more than 0.77 as long as $m/n \leq 0.5$.

7.3 Feedback Recovery Efficiency of Our Protocol

In this subsection, we analyze the relationship between m/n and the efficiency of feedback recovery, under different percentages of selfish raters. We evaluate the efficiency

of feedback recovery as the number of iterations required to obtain the secret key shares during the enquiry phase. An enquirer can progressively ask raters for the decrypted secret key shares. Ideally, only one iteration is required if m raters reply to the request. If some raters are selfish, more iterations are needed until m replies are collected to reconstruct the feedback sum. Therefore, the number of iterations required indicate the efficiency. In the simulations, the number of iterations required to recover the feedback sum of each CSP is determined so that the overall average can be computed. Similar to Section 7.2, we randomly choose selfish raters $\in \{20\%, 30\%, 40\%, 50\%\}$ of the entire network when running the simulation in the trust graph. For the random graphs, we assume 50% selfish raters.

We present the recovery efficiency of our protocol in Fig. 9 and Fig. 10. *Advogato* and *Random-a* show the similar trend in Fig. 9 (a) and Fig. 10 (a). Whereas, the peak number of iterations required in *Random-b* is much higher than that in the other three graphs as shown in Fig. 10 (b). Comparing the result of *Random-a* and *Random-b*, it can be seen that the network density p has more influence on the feedback recovery efficiency. When the network density $p \geq 10\%$, the enquirer needs more iterations to obtain the feedback sum (i.e., less efficient). The decline shown in the figures indicates that the trust result of some CSPs cannot be obtained when the number of selfish raters and m/n is large as discussed in Section 7.2. By considering the Intercloud configuration and security requirements, a suitable threshold value of m can be chosen.

7.4 Trust Result Accuracy Comparison

In this subsection, we compare the trust result accuracy of our protocol with the scheme that uses additive secret sharing (ASS). Clark et al. [30] and Hasan et al. [19] both used ASS in their schemes to protect feedback privacy. Given a secret f , an ASS of f consists of n shares $\{f_1, \dots, f_n\}$, where f_1, \dots, f_{n-1} are randomly chosen and $f_n = f - \sum_{i=1}^{n-1} f_i$. To accurately recover the secret, it simply adds all of the secret shares together.

ASS requires all raters to stay online to enable the computation of trust results. When any rater leaves the network or refuses to reply, all corresponding feedback shares will be lost in the trust result, causing inaccurate trust results. To perform the analysis, let \mathcal{T}_r denote the trust result of each CSP computed from all of its raters. Let \mathcal{T}_r' denote the trust result of each CSP computed from the raters who are not selfish. We consider the trust result accuracy as the average value $\frac{\mathcal{T}_r'}{\mathcal{T}_r}$ of all CSPs whose trust results can be recovered. In the subsequent simulations, we randomly choose selfish raters $\in [5\%, 50\%]$, with an increment of 5% in the *Advogato* trust graph. We set $m/n = v/n = 0.4$ in our protocol and in ASS approach, such that the majority of feedback can be recovered (v is the number of feedback shares prepared by each rater in Hasan et al.'s scheme).

Fig. 11 shows the comparison of trust result accuracy on the *Advogato* trust graph. We observe that the accuracy of the ASS approach decreases with an increased percentage of selfish raters. However, the accuracy of our protocol can be maintained at a high level. This is because in our protocol, an enquirer only needs to request $m < n$ raters to

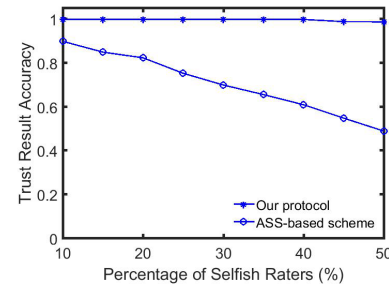


Fig. 11: Trust result accuracy comparison of our protocol and ASS-based scheme.

reconstruct an accurate trust result. Therefore, our protocol is less affected by the percentage of selfish raters, unless the percentage is unreasonably high.

8 RELATED WORK

Recent Intercloud research has mainly focused on resource management (e.g., an adaptive cloud resource allocation scheme [36], a simulation framework for Intercloud job scheduling [7]), data and virtual machine migration (e.g., a software defined network-based Intercloud virtual machine migration scheme [9], a decision-making model for Intercloud migration from the user perspective [10]), service integration (e.g., an algorithm for selecting and composing services among multiple clouds [6], a mechanism to interconnect virtual machines at different clouds [8]) and Intercloud security (e.g., a single sign-on authentication scheme for clouds to inter-operate with one another [37], an authentication mechanism for components running at different clouds [38]). Complementing the aforementioned research work, there is a need to study trust management (e.g., reputation-based trust management) for Intercloud.

For trust management, various reputation-based trust management protocols with privacy protection have been proposed for different purposes. For instance, Clark et al. proposed a reputation system with privacy protection for mobile ad-hoc networks [30] with the focus on handling dynamic configuration (i.e., parties may join and leave dynamically). Since each party in Clark et al.'s scheme shares its feedback with all other participants, the complexity of message exchange in Clark et al.'s scheme is $O(n^2)$. Schaub et al. [34] proposed an anonymous reputation system for e-commerce websites, with the aim of guaranteeing user anonymity in the trust evaluation process. However, feedback privacy cannot be protected. Tormo et al. proposed a reputation management system for hybrid broadcast broadband TV (HbbTV) [35], which aims to compute personalized trust results based on the similarity of two users when making choices. However, the scheme needs to disclose both feedback and user identities to trusted third parties. Hasan et al. designed a privacy-preserving reputation protocol that can tackle attacks by malicious participants in a multi-agent environment [19]. However, this scheme leaks the existing trust relationships between users of the same CSP. And it cannot compute customized evaluation results and ensure user anonymity. Although some of these protocols may be adapted for Intercloud with some modifications, it is still

TABLE 4: Comparison of trust-related protocols

System/Protocol	Core architecture	Scenario	Customized evaluation result	Feedback privacy	User anonymity
Bernstein and Vij [31]	Centralized	Intercloud	No	No	No
Abawajy [32]	Centralized	Intercloud	No	No	No
Ngo et al. [33]	Distributed	Intercloud	No	No	No
Clark et al. [30]	Distributed	Ad-hoc network	No	Yes	No
Schaub et al. [34]	Distributed	E-commerce	No	No	Yes
Tormo et al. [35]	Centralized	HbbTV	Weighted mean	No	No
Hasan et al. [19]	Distributed	Multi-agent	No	Yes	No
Our protocol	Distributed	Intercloud	Weighted mean	Yes	Yes

desirable to develop an Intercloud-specific trust evaluation protocol with the advent of cloud computing, as well as for effectiveness and efficiency considerations.

Previous Intercloud trust management frameworks were mainly Public Key Infrastructure (PKI)-based, reputation-based and attribute-based. Bernstein and Vij presented a PKI-based trust model for Intercloud, relying on the use of a global “Trust Index” under a centralized architecture [31]. Apart from security, scalability may be a concern. Abawajy presented a reputation-based trust model for Intercloud by employing a reputation manager to collect trust ratings and computing the reputation of all clouds under a centralized architecture [32]. However, feedback privacy cannot be protected. Ngo et al. proposed an Intercloud trust model based on the attribute-based access control policy [33]. Trust relationships are built on the basis of context (i.e., a set of attributes) and the trust criteria can be formulated as a logical expression of these attributes. While indirect trust relationships can be built through recommendations between clouds in a distributed manner, the chain of trust will be broken when any one of the intermediate clouds is offline or is controlled by an attacker. Again, there is no privacy protection for feedback.

Table 4 compares different trust-related protocols in terms of different attributes. Our protocol has distinctive advantages. Apart from using a distributed architecture, our protocol can still accurately compute the trust result, even when some of the users are unavailable. This is an important requirement for a distributed system or dynamic network. For example, Zhu et al. studied a probabilistic misbehavior detection scheme toward efficient trust evaluation in delay/disruption tolerant networks (DTNs) [39]. As highlighted by [39], malicious nodes (i.e., drop or modify messages) and selfish nodes (i.e., refuse to forward messages) seriously affect trust evaluation and network performance. As shown by the aforementioned simulation results, our protocol can handle this kind of attack effectively. Our protocol can compute customized trust evaluation results while protecting feedback privacy. However, unlike [19], [30], our protocol employs a verifiable secret sharing scheme to facilitate flexible processing and feedback updating. Our protocol also supports user anonymity during the trust evaluation process. Compared to [19] and [30], our protocol is also more efficient in terms of message transfer complexity during the enquiry phase. Note that during the enquiry/query phase, an enquirer in Hasan et al.’s scheme [19] needs to forward

vn encrypted secret shares and $vn + n$ verifiable values between raters, where $v \ll n$ is the number of shares prepared by each rater. Furthermore, raters need to reply to the enquirer with n sum of shares for aggregation. Thus, in total, Hasan et al.’s scheme needs to transmit $2vn + 2n$ (i.e., $O(n)$) messages. Our protocol requires downloading m encrypted secret shares and m commitments for verification from FBS ($m < n$ is the threshold value for secret key reconstruction). Furthermore, raters need to reply to the enquirer with m sum of shares for secret key reconstruction. In total, our protocol needs to exchange $3m$ (i.e., $O(m)$) messages. Hence, our protocol is more efficient.

9 CONCLUSION AND FUTURE WORK

In conclusion, we have presented a distributed trust evaluation protocol with privacy protection for Intercloud. Compared to other protocols, this distributed protocol provides some distinctive features, particularly for the Intercloud environment. First, it supports user anonymity by means of blind signature, facilitating users to provide honest feedback without fear of a retaliatory attack. Second, by means of an innovative mechanism for storing feedback, feedback privacy can be protected by using homomorphic encryption with verifiable secret sharing. Third, it allows customized processing of evaluation results while protecting feedback privacy. A security model has been employed to evaluate the protocol for its effectiveness. Unlike many other distributed protocols, which only support static configuration, the protocol can still be effective when some of the parties are offline (i.e., supporting a dynamic configuration). Simulation results indicate the protocol can still function well when half of the parties are malicious or offline. Future work is being planned to further analyze and enhance the protocol (e.g., using distributed ledger technology). For example, various blockchains can be formed (e.g., among Intercloud Exchanges, CSPs and users). It is of interest to study how the blockchains can interact to support trust evaluation and other advanced functions for Intercloud.

ACKNOWLEDGMENT

This work is supported by the Department of Computing, The Hong Kong Polytechnic University.

REFERENCES

- [1] K. Kaur, S. Sharma, and K. S. Kahlon, "Interoperability and portability approaches in inter-connected clouds: A review," *ACM Comput. Surv.*, vol. 50, no. 4, pp. 49:1–49:40, 2017.
- [2] A. J. Ferrer, "Inter-cloud research: Vision for 2020," in *2nd International Conference on Cloud Forward: From Distributed to Complete Computing, Madrid, Spain, 18-20 October, 2016.*, 2016, pp. 140–143.
- [3] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective," *J. Cloud Computing*, vol. 5, p. 4, 2016.
- [4] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Comput. Surv.*, vol. 47, no. 1, p. 7, 2014.
- [5] T. Truong-Huu and C.-K. Tham, "A novel model for competition and cooperation among cloud providers," *IEEE Trans. on Cloud Comput.*, vol. 2, no. 3, pp. 251–265, 2014.
- [6] L. Liu, S. Gu, D. Fu, M. Zhang, and R. Buyya, "A new multi-objective evolutionary algorithm for inter-cloud service composition," *TIIS*, vol. 12, no. 1, pp. 1–20, 2018.
- [7] S. Sotiriadis, N. Bessis, A. Anjum, and R. Buyya, "An inter-cloud meta-scheduling (ICMS) simulation framework: Architecture and evaluation," *IEEE Trans. Services Computing*, vol. 11, no. 1, pp. 5–19, 2018.
- [8] L. Osmari, S. Toor, M. Komu, M. J. Kortelainen, T. Lindén, J. White, R. H. Khan, P. Eerola, and S. Tarkoma, "Secure cloud connectivity for scientific applications," *IEEE Trans. Services Computing*, vol. 11, no. 4, pp. 658–670, 2018.
- [9] S. Sotiriadis, N. Bessis, E. G. Petrakis, C. Amza, C. Negru, and M. Mocanu, "Virtual machine cluster mobility in inter-cloud platforms," *Future Generation Comp. Syst.*, vol. 74, pp. 179–189, 2017.
- [10] E. Barlasakar, P. Kilpatrick, I. T. A. Spence, and D. S. Nikolopoulos, "Myminder: A user-centric decision making framework for intercloud migration," in *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, April 24-26, 2017.*, 2017, pp. 560–567.
- [11] S. Sotiriadis and N. Bessis, "An inter-cloud bridge system for heterogeneous cloud platforms," *Future Generation Comp. Syst.*, vol. 54, pp. 180–194, 2016.
- [12] (2018, May) Cisco intercloud fabric. [Online]. Available: <https://www.cisco.com/c/en/us/products/cloud-systems-management/intercloud-fabric/index.html>
- [13] D. Bernstein and D. Vij, *IEEE PROJECT 2302 - Standard for Intercloud Interoperability and Federation (SIIF)*, IEEE Standards Association Department Std., 2012. [Online]. Available: <https://standards.ieee.org/develop/project/2302.html>
- [14] C. K. Law, W. Xie, Z. Xu, Y. Dou, C. T. Yu, H. C. B. Chan, and D. W. K. Kwong, "System and protocols for secure intercloud communications," in *11th International Conference for Internet Technology and Secured Transactions, ICITST 2016, Barcelona, Spain, December 5-7, 2016*, 2016, pp. 399–404.
- [15] C. T. Yu, H. C. B. Chan, and D. W. K. Kwong, "Discovering resources in an intercloud environment," in *2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017*, 2017, pp. 1–6.
- [16] Y. H. Ho, P. M. F. Ho, and H. C. B. Chan, "Mobile intercloud system and objects transfer mechanism," in *2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017*, 2017, pp. 1–6.
- [17] T. H. Noor, Q. Z. Sheng, S. Zeadally, and J. Yu, "Trust management of services in cloud environments: Obstacles and solutions," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 12:1–12:30, Jul. 2013.
- [18] T. H. Noor, Q. Z. Sheng, L. Yao, S. Dustdar, and A. H. Ngu, "Cloudarmor: Supporting reputation-based trust management for cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 367–380, 2016.
- [19] O. Hasan, L. Brunie, E. Bertino, and N. Shang, "A decentralized privacy preserving reputation protocol for the malicious adversarial model," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 6, pp. 949–962, 2013.
- [20] S. Tadelis, "The economics of reputation and feedback systems in e-commerce marketplaces," *IEEE Internet Computing*, vol. 20, no. 1, pp. 12–19, 2016.
- [21] O. Adewoyin, R. Araya, and J. Vassileva, "Peer review in mentorship: Perception of the helpfulness of review and reciprocal ratings," in *Intelligent Tutoring Systems - 13th International Conference, ITS 2016, Zagreb, Croatia, June 7-10, 2016. Proceedings*, 2016, pp. 286–293.
- [22] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *TOSN*, vol. 5, no. 3, pp. 20:1–20:36, 2009.
- [23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, 1999, pp. 223–238.
- [24] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, 1987, pp. 427–437.
- [25] T. Okamoto, "Efficient blind and partially blind signatures without random oracles," in *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, 2006, pp. 80–99.
- [26] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [27] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 251–260.
- [28] Y. Lindell, "How to simulate it - A tutorial on the simulation proof technique," *IACR Cryptology ePrint Archive*, vol. 2016, p. 46, 2016.
- [29] M. H. Au, P. P. Tsang, and A. Kapadia, "Perea: Practical ttp-free revocation of repeatedly misbehaving anonymous users," *ACM Trans. Inf. Syst. Security*, vol. 14, no. 4, pp. 29:1–29:34, Dec 2008.
- [30] M. R. Clark, K. E. Stewart, and K. M. Hopkinson, "Dynamic, privacy-preserving decentralized reputation systems," *IEEE Trans. Mob. Comput.*, vol. 16, no. 9, pp. 2506–2517, 2017.
- [31] D. Bernstein and D. Vij, "Intercloud exchanges and roots topology and trust blueprint," in *Proc. 11th Int. Conf. Internet Comput.*, 2011, pp. 135–141.
- [32] J. Abawajy, "Determining service trustworthiness in intercloud computing environments," in *Proc. 10th Int. Symp. Pervasive Syst., Algorithms, and Networks*, 2009, pp. 784–788.
- [33] C. Ngo, Y. Demchenko, and C. de Laat, "Toward a dynamic trust establishment approach for multi-provider intercloud environment," in *Proc. 4th Int. Conf. Cloud Comput. Technol. and Sci.*, 2012, pp. 532–538.
- [34] A. Schaub, R. Bazin, O. Hasan, and L. Brunie, "A trustless privacy-preserving reputation system," *IACR Cryptology ePrint Archive*, vol. 2016, p. 16, 2016.
- [35] G. D. Tormo, F. G. Mármol, and G. M. Pérez, "Towards privacy-preserving reputation management for hybrid broadcast broadband applications," *Computers & Security*, vol. 49, pp. 220–238, 2015.
- [36] B. Chang, Y. Lee, and Y. Liang, "Reward-based markov chain analysis adaptive global resource management for inter-cloud computing," *Future Generation Comp. Syst.*, vol. 79, pp. 588–603, 2018.
- [37] C. Powell, T. Aizawa, and M. Munetomo, "Design of an sso authentication infrastructure for heterogeneous inter-cloud environments," in *Pro. 3rd Int. Conf. Cloud Netw.*, 2014, pp. 102–107.
- [38] K. Walsh and J. Manferdelli, "Intra-cloud and inter-cloud authentication," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, June 25-30, 2017*, 2017, pp. 318–325.
- [39] H. Zhu, S. Du, Z. Gao, M. Dong, and Z. Cao, "A probabilistic misbehavior detection scheme toward efficient trust establishment in delay-tolerant networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 22–32, 2014.

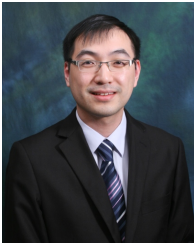


Yi Dou received the B.Eng. degree in Computer Science and Technology (Information Security) and M.Eng. degree in Computer Software and Theory from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2011 and 2014, respectively, and the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, in 2018. Her research interests include information security, cloud computing and networking.



Henry C. B. Chan received the B.A. and M.A. degrees from the University of Cambridge, United Kingdom, and the Ph.D. degree from the University of British Columbia, Canada. He is currently an associate professor and associate head of the Department of Computing, The Hong Kong Polytechnic University (PolyU). His research interests include networking/communications, cloud computing, electronic commerce, and computing education. He was the recipient of the 2015 IEEE Computer

Society's Computer Science and Engineering Undergraduate Teaching Award for his outstanding contributions to computing education through teaching, mentoring students, and service to the education community. He received four President's awards from PolyU. He was the chair of the IEEE Hong Kong Section in 2012, and the IEEE Hong Kong Section Computer Society Chapter from 2008 to 2009.



Man Ho Au received his Ph.D. degree from the University of Wollongong in 2009. He is an assistant professor and a director of the Monash-PolyU-CC Joint Lab on Blockchain and Cryptocurrency Technologies at the Department of Computing, The Hong Kong Polytechnic University. His research interests include information security, applied cryptography and blockchain technology. He has published over 140 refereed papers in reputable journal and conferences, including ACM CCS, ACM SIGMOD, NDSS, IEEE

TIFS, TC, TKDE, etc. He is a recipient of the 2009 PET runner-up award for outstanding research in privacy enhancing technologies. He received best paper awards from conferences including ACISP 2016 and ISPEC 2017. According to Google Scholar, his h-index is 33 and his work has been cited over 3400 times. He is an expert member of the ISO/IEC JTC 1/SC 27 working group 2 - Cryptography and security mechanisms and a committee member of the Hong Kong Blockchain Society R&D division.