Enhancing Trust Management in Cloud Computing

Abdelmageed Algamdi

Department of Computer Science

University of Liverpool

This dissertation is submitted for the degree of

Doctor of Philosophy

November 2017

I would like to dedicate this thesis to my loving parents, my wife and my only son.

Acknowledgements

First, I would like to thank my supervisors, Frans Coenen and Alexei Lisitsa, for all of their advice and encouragement. It was wonderful working with them on the cloud trust management system. I extend my thanks to them for providing guidance in my research over the past three years. During the first few months of my PhD research, while I was chasing half-baked ideas, Alexei was very patient with me and then gently suggested a series of ideas which actually worked. Both Frans and Alexei provided lots of help in writing and revising my research papers.

I would also like to thank my advisors, Boris Konev and Andre Hernich, for providing much encouragement, support, feedback and advice during my three years of study.

I am grateful to my wife for providing support and empowering me to succeed, and I extend thanks to my son, Hossam, who gave me inspiration, through his smiles, to continue doing my PhD.

Thank you to my family, especially my parents, for supporting me throughout my entire life.

Finally, I would like to thank the Saudi government for funding my PhD.

Abstract

In this thesis, the issue of trust in cloud computing is addressed. We propose a novel architecture for a trust management system for cloud computing in which various sources of trust-related information are utilised and different trust mechanisms are combined. Through this architecture, cloud users can utilise typically distributed cloud trust protocol (CTP) to ask questions about cloud services and receive answers via a hypertext transfer protocol (HTTP)response. We introduce cloud user assessments and their importance, which complements the self-assessments of providersdone in previous work. We use questionnaires – the Consensus Assessment Initiative Questionnaire (CAIQ) and a Smals research group questionnaire – to get opinions from providers and users, respectively. The providers' opinions reflect the service capabilities claimed by the provider, while the users' opinions reflect their satisfaction with the service being offered over the cloud.

Based on the providers' and users' opinions, we propose trust aggregation, ageing and reputation mechanisms to generate digital trust values for services. Trust-related information is presented and processed in terms of opinions formalised in subjective logic. We represent the providers' and users' opinions as binomial subjective opinions that express not only the beliefs but also the uncertainties found in the questionnaire answers. We also present a classification method for subjective opinions based on barycentric coordinates that classify binomial subjective opinions into rating classes, which can be defined as scalar rating values.

We also introduce the concept of trust transfer between users for calculating their level of trust. The trust transfer is a mechanism by which old users can share their opinions regarding a certain service with new users on demand. We give new users the ability to assess the service themselves or to ask other agents – old users who assessed the service before – to provide a recommendation based on their opinions. The recommendation requests will be represented as a directed tree; the top parent (i.e., the top of the tree) represents the new users, while the leaves represent old users, who have completed their assessments without help. The other nodes represent old users who have

transferred opinions to their child nodes (i.e., other users who trust them). We propose a mechanism for opinion transferring among cloud users based on cumulative, averaging and transitivity fusion logic operations.

We implemented a graphical user interface (GUI) tool using Java to conduct the provider and user assessments, from collecting their opinions using questionnaires to generating the overall trust values for the services. We provide calculations using two methods: aggregation only and aggregation with ageing. We performed extensive experiments to find the best method that can be used for collecting opinions from both providers and users. We present the experimental results and the system evaluation.

The last contribution of this thesis is to verify trust in the users themselves before including their opinions in the trust reputation system. We use blockchain technology to achieve this. We provide a voting system that can be implemented between trusted users to allow them to accept or reject a new user inside a trusted area. Once a new user gains the majority of user votes, his or her opinion towards the service affects the service trust value; otherwise, the user's opinion will be discarded. The benefit of using blockchain technology is to provide decentralised decisions without any interference from other members. We also introduce important constraints for the voting process that preserve fair and secured decisions.

Contents

List of Figures

List of Tables

Listings

Notation

Symbol	Description
$\mathbb{X}$	represents the subjective logic domain
$\rho(\mathbb{X})$	represents the subjective logic hyperdomain
ω_x	represents a subjective opinion about service x
ω_x^A	represents the subjective opinion of agent A about service x
$\omega_{(x_1 \cup x_2)}$	represents the subjective logic addition between ω_{x_1} and ω_{x_2}
$\omega_{(x_1 \wedge x_2)}$	represents the subjective logic between ω_{x_1} and ω_{x_2}
$\omega_x^{A \diamond B}$ and $\omega_x^A \oplus \omega_x^B$	represents the subjective logic consensus between two independent opinions, ω_x^A and ω_x^B
$\omega_x^{A_1} \underline{\oplus} ... \underline{\oplus} \omega_x^{A_n}$	represents the averaging fusion operation between opinions $\omega_x^{A_i}$, where $i \in [1, n]$
$\cup$	is an addition operation in subjective logic
$\wedge$	is an and/product operation in subjective logic
$\oplus$	is a consensus operation in subjective logic
$\underline{\oplus}$	is an averaging operation in subjective logic
b_x	represents the belief aspect of a subjective opinion, ω_x
d_x	represents the disbelief aspect of a subjective opinion, ω_x
u_x	represents the uncertainty aspect of a subjective opinion, ω_x
a_x	represents the base rate of a subjective opinion, ω_x
P_x	represents the projected probability of a subjective opinion, ω_x
k	represents the scalar value corresponding to the subjective opinion rating class
k_{t+1}	represents the scalar value corresponding to the subjective opinion rating class at time $(t+1)$, always referring to the most recent opinion class
k_t	represents the scalar value corresponding to the subjective opinion rating class at time (t), always referring to the oldest opinion class
K	represents a set of n opinion rating classes

Symbol	Description
o	represents the opinions presented in (22)
t	represents the average rating for the opinions presented in (22)
c	represents the level of certainty for the opinions presented in (22)
f	represents the initial expectation for the trust value of a given service (22)
$\omega_x^{A(i,j)}$	represents the binomial opinion of sub-attribute j within attribute i, generated from cloud user A's assessment of service x
$R(i)$	represents a relation that generates the sub-attributes within attribute i
$\omega_x^{A(i)}$	represents the binomial opinion of attribute i, generated from cloud user A's assessment of service x
λ	is the aggregation constant
Λ	is the ageing factor
$\mathbb{A}$	represents the set of all users who completed assessments for service x
$R_{x,(t=0)}$	is the initial rating value (i.e., of the provider only) generated from the provider's self-assessment for service x at the beginning ($t = 0$)
$R^A_{x,(t\neq 0)}$	is the old rating value (i.e., of the provider and user A) over time t for service x
$R^A_{x,(t+1)}$	represents the overall new accumulated rating value (i.e., of the provider and user A) after time period $t+1$ for service x
$R_{x,(t+1)}$	represents the overall new accumulated rating value (i.e., of the provider and all users) after time period $t+1$ for service x
$\delta^A_{x,t+1}$	represents the effect of the most recent update at time $t+1$ for user A in the assessment of service x
$\delta^A_{x,t}$	represents the effect of the update for all historical assessments in the time period of $[0, t]$ for user A in the assessment of service x
$R^*_{x,(t+1)}$	represents the estimation of the final trust value at time $(t+1)$ for the testing procedure

Symbol	Description
$\eta\omega_x^{A\triangledown B}$	represents the overall weighted opinion of both agents A and B towards service x with overall weight η
$\boxplus$	is the weighted consensus operation
α_i	is the weight of opinion $\omega_x^{A_i}$
$\omega_x^{A:B}$	represents an opinion transferred from agent B to agent A towards predicate x using transitivity
$\otimes$	represents a transitivity operation
e	is the absolute error between the calculated trust in a program and the estimated trust of a tester for a single test iteration
e_λ	is the absolute error between the calculated trust in a program and the estimated trust of a tester at a specific λ value
$\overline{e_\lambda}$	is the average absolute error between the calculated trust in a program and the estimated trust of a tester at a specific λ value
e_Λ	is the absolute error between the calculated trust in a program and the estimated trust of a tester at a specific Λ value
$\overline{e_\Lambda}$	is the average absolute error between the calculated trust in a program and the estimated trust of a tester at a specific Λ value
$\mathbb{N}$	represents a natural number

Term	Meaning
CTP	cloud trust protocol
MCQ	multiple choice questionnaire
CAIQ	Consensus Assessment Initiative Questionnaire
GUI	graphical user interface
CPU	central processing unit
QoS	quality of service
TR	trust and reputation
SLA	service level agreement
RESTful	Representational State Transfer API
HTTP	hypertext transfer protocol
EOT	element of transparency
URL	universal resource locator
GTS	general trust server
LTS	local trust server
RqMg	request manager
IAM	identity and access management

Chapter 1

Introduction

In recent years, cloud computing has come to be considered an important technology, allowing users to remotely share various resources over the internet. Through virtualisation and job scheduling, cloud computing can be employed in a unified manner. Due to the increasing importance of cloud computing in our daily lives, extensive research has been done to enhance its capabilities and ensure that everythingis running according to standard.

1.1 Cloud and Trust

An important area of study is achieving trust between cloud providers and customers. Hence, before introducing this problem, we will present an introduction to cloud computing.

1.1.1 Cloud

Three types of cloud computing service models are widely used: software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) (7), (46). In an SaaS model, the software and application are installed and run by cloud providers on a cloud platform and infrastructure. Access to the software is provided to cloud users, but they have no permission to maintain or manage the software and application. SaaS has the least freedom of operation. In a PaaS model, cloud providers offer a computer platform, involving a programming execution environment, database, website and operation system. Cloud users can install or develop software on the cloud platform, and they do not have to purchase the high-cost and complex underlying hardware and

software layers. PaaS offers mid-level freedom of operation. In an IaaS model, cloud providers deliver computer infrastructure resources, such as a virtual machine. Cloud users can choose the configuration of the virtual machine based on their preference, including the type and number of central processing unit (CPU), the size of hard disk and memory, the firewall, the network and the sound device used. Users deploy the platform or operation system integrated into the virtual machine and can then install the software and application on the platform. IaaS has the most freedom of operation. Three typical application cases of SaaS, PaaS and IaaS are Microsoft Office Online, Google App Engine and VMware, respectively (7; 46).

Despite the numerous advantages of using the cloud, cloud users may have some concerns regarding how to control their data and how to make sure that no one can access it except the owner. Another issue is availability, since online services are bound to have downtime, and, therefore, data may not be available when the user needs them. Hence, trust needs to be built between the customers and the providers offering cloud services.

1.1.2 Trust

Many definitions of trust have been developed in social science (10; 45). Merriam-Websterprovides a simple definition of trust, which is 'a belief that someone or something is reliable, good, honest, effective, etc. or to have confidence in someone or something' (47). It can also be defined as a mental state comprising expectancy, belief and willingness to take risk on behalf of the trustor, the consumer, and the trustee, the cloud service provider (24).

In computer science, trust is known as computational trust, which can be accomplished using cryptography. Computational trust reflects the user's evidence-based opinion towards computations done by a program. There are various trust models that control the trustworthiness calculations based on such evidence. The existence of various trust models has encouraged developers to increase the reliability and performance of their digital products. Trust can be computed based on conceptual models, information sources, direct experiences and witness information. The most common models are those based on direct information, which indicates experience based on direct interaction between a user and a digital system (56).

We can say that an entity, A (a trustor), trusts another entity, B (a trustee), if the following three statements are satisfied.

1. A expects a certain **behaviour** from B, which is known as **expectancy**.

2. Entity A believes that the expected behaviour will be demonstrated in an acceptable manner, which is known as **belief**. The belief is always built on evidence seen by the trustor.

3. The trustor, A, is **willing to take a risk** for his or her belief.

*Trust versus reputation.*Trust is different from reputation. We can characterise trust as the 'oxygen' that permits the existence of reputation. Trust involves a relationship between two entities, while reputation is a value generated by aggregating a community's opinion towards a certain entity.

Trust in a cloud environment. In a cloud environment, trust encompasses and relies upon diverse attributes, such as information security, compliance and data governance. So, potential users have to identify credible clouding software with required specifications. For cloud computing, trust management monitors the quality of service (QoS) and the service level, and there is a contract between the cloud user and service provider; hence, verification is an important field (68). The overall view of a community towards a cloud service provider can be expressed by their reputation. In this project, we build a system to measure the reputation of cloud service providers. This is done using the trust information acquired by the cloud trust protocol (CTP) (38; 39; 42; 43) and the feedback of users who have already used services offered by the provider to assess the service quality using the Consensus Assessments Initiative Questionnaire (CAIQ) (18).

An extension of the definition of trust in cloud computing, presented above, is that the trustor (entity A) can be any cloud user, while the trustee (entity B) represents a cloud provider or a service. The cloud user expects a certain behaviour from the provider regarding how the service will run, and belief is instilled in the cloud user after they encounter evidence that the provider is fulfilling their expectation. Users can get help from transparency protocols, such as the CTP, to ask and receive answers about issues regarding the service performance, or they can get help from a reputation system that is managed by experts, in which each service or provider is assigned a value that represents its reputation. The higher the value, the better reputation. Finally, the cloud user shows his or her willingness to take a risk by beginning to use the service.

Cloud trust management systems (21; 22; 32; 60) are responsible for calculating trustworthiness and finding trustworthy services. This is done based on trust and reputation (TR) models, which translate the nature of different attributes, such as data governance, compliance to the regulations and information security, into a value. TR

systems are an example of how to build trust in various service environments. Although these systems provide useful TR models for decision making, most of them fail to consider multiple attributes, such as security, compliance and data governance (32).

Trust types. Based on trustor expectancy, Huang and Nicol presented two types of trust: (1) trust in performance and (2) trust in belief (25; 26).

- Trust in performance.

  We can represent the trust relation between two entities (A as a trustor and B as a trustee) by the relation $trust_p(A, B, x, k)$ if A believes the performance of x in the context k, where x is performed by B in the context k. This means that if x is made by B in a context k, then A believes x in that context (25; 26).

- Trust in belief. A trust relation is denoted by $trust_b(A, B, x, k)$. This relation is satisfied if A trusts B regarding the belief of x in a context k. This means that if B believes x in a context k, then A also believes x in that context (25; 26).

Trust as a result of assessments. As we said before, the trust established between entities (i.e., users and providers) affects the reputation of trustee entities. Hence, the reputation of a cloud provider or a service can be calculated by aggregating the cloud users' opinions. There are many ways to ask users for their opinions. Amazon, BizRate and eBay ask their users to rate the products they buy. Amazon builds trust in a certain product by averaging the buyer ratings, while eBay builds trust based on the percentage of positive feedback for an item. Each merchant has a value that represents the average level of satisfaction with the items sold by the merchant. On BizRate, the level of trust in an item is calculated by compiling the average satisfaction factor of the seller from the buyer reviews for that item (20). As a cloud service (similar to a mailing service) must meet various requirements, including privacy, security and governance, the users' opinions must reflect all of these aspects. Therefore, we will use a multiple choice questionnaire (MCQ) that asks about all aspects related to the services, taking an intelligent approach to build an overall opinion that reflects all of these aspects.

1.2 CTP and CAIQ.

The role of transparency is acknowledged by the development of the CTP (38; 39). This high-level protocol is meant to achieve cloud providers with transparency via

a query–response mechanism, allowing (potential) users to query providers about trust-related information. Starting with a high-level specification (38; 39), the protocol has recently attained the proposed application programming interface (API)(50), which brings it closer to the implementation stage. Building upon these developments, in this paper, we propose a novel architecture for a cloud trust management system in which various sources of trust-related information are utilised and different trust mechanisms are combined. This includes using distributed CTP, CAIQ, trust aggregation and reputation systems. To assess the service quality, the CAIQ assessment is used based on the trust information acquired by the CTP and from the feedback of those who previously used the services offered by the cloud service provider.

Consumer assessments are also useful to produce a fair evaluation of trustworthiness, as the system could be subject to attacks. The ability of any user to do an assessment and modify the trustworthiness generated by the self-assessment operation could make the system vulnerable. Unfair user opinions coming from malicious (untrusted) cloud users could update the trustworthiness with wrong values, so the latest trustworthiness value could be incorrect, failing to reflect the trust relation between the provider and the average cloud user accurately. Untrusted users' assessments should be blocked from updating the trust value, and the decision to block a user should be made intelligently so that no good opinion is overlooked. There are different ways to do this, but the most logical and fair way is to give a blocking decision based on a democratic voting process. We will use blockchain technology to build a decentralised voting system that decides whether a new user's assessments will be accepted, and permitted to update the trust, or rejected. The voting system includes all previous users that have assessed the service before as members. Once a new user submits his or her assessment, this user appears as a candidate, and the members can either accept or reject his or her assessment. The members can review the questionnaire that was completed by the new user and see their answers. In order to encourage previous users to review new users' assessments, and to do so fairly, a reward/penalty approach is proposed.

1.3 Motivation and Objectives

1.3.1 Motivation

Users, myself included, are motivated by many applications that present their services to us. As an example, let us discuss my personal experience with a food delivery application called Just Eat.Just Eat is one of the applications that can be used to order food from a restaurant in the user's vicinity. The first time I tried to use it to order food, I did not recognize any of the restaurants that appeared in the application. I just ordered my favourite meal from one of the restaurants at random, but unfortunately, the food quality was poor. Later, I received a survey from the application to assess the food quality, and of course, I gave the restaurant only one star out of five. When I tried to order food a second time, I sorted the restaurants by their rating. I selected a restaurant that had a high rating from numerous clients, and, as expected, the food was quite good.

Motivated by many similar situations to the one I mentioned above, I thought about building a TR system for services offered over the cloud. The proposed TR system reflects not only the providers' opinions about their services but also the customers' opinions after using these services. We simply ask the users to assess the services offered over the cloud, based on their experiences. The most common feedback method used online is asking the customer to provide a scalar value, or rating, which represents the customer's satisfaction level. For food services, it may be sufficient to ask the user directly about their satisfaction level, but for other services offered over the cloud, similar to a mailing service, it would be unfair to ask the user to submit his or her satisfaction level directly without going into more detail regarding the service specifications. Accordingly, we have used a descriptive MCQ that asks the user about individual features the service provides, without asking them to enter their satisfaction level directly.A user's opinion can be obtained fairly from his or her answers. We then begin collecting the opinions of all the users to produce indicator values for how the services fit the users' desires. These measurements can be seen by anyone over the cloud as well as by the cloud providers, which creates competition between providers, motivating them to enhance their services so that they will receive high ratings and attract more clients.

However, there are potential problems with the process of giving cloud users the ability to express how much they trust the provider because the resulting values affect the overall trust value of the provider. The overall system could be vulnerable to security attacks from malicious assessments, which change the overall trust values unfairly. Consequently, a filtering process is needed before updating the overall trust value for

the cloud provider, and this process should be done in a decentralised manner in which no single node influences the decision.

1.3.2 Objectives

The TR systems implemented in the past reflect only the provider's point of view; the trust values generated for these systems do not reflect the users' satisfaction about the services being offered over the cloud. Therefore, our main research questions are as follows: How can we enhance the calculation of trust so that it reflects the service capability as well as the user satisfaction with that service? What infrastructure can be used to achieve that aim? As we aim to give all cloud users the ability to assess any service offered over the cloud and the ability to update the trust values generated by TR systems in a way that reflects their satisfaction levels, we need to find a way to remove unfair assessments created by cloud users before updating the TR system.

1.4 Contributions and Thesis Outline

1.4.1 Contributions

The overall contribution of this thesis is to present a TR system that reflects the provider's opinions as well as the cloud users' opinions about services being offered over the cloud. The proposed TR system should take the users' opinions into account in a secure and fair manner.

We present an infrastructure for a system which provides us with the ability to use the CTP, ask for assessments, calculate the digital trust for providers and ask for queries based on stored trust values. The suggested infrastructure not only enables providers to complete the self-assessment process but also gives cloud consumers the opportunity to reassess a service at any time. This consumer assessment reflects the satisfaction of the user, which is the main factor affecting the digital trust value.This is achieved using an MCQ designed for both cloud providers and cloud consumers, allowing them to complete the two assessments mentioned above. The assessment process is based on the provider's and consumers' answers to these MCQs.Provider and user opinions are extracted from their answers to these MCQs using subjective logic operators, such as multiplication, consensus and averaging operators (31), which are applied to binomial opinions represented by quadruples of real values, each within an interval of $[0 \dots 1]$. The provider's opinion creates the initial trust value for a service, while a consumer's opinion

updates that value, depending on his or her satisfaction level with the service. This is also done using subjective logic operators to aggregate the opinions – using an aggregation constant – of different users, taking into account the timing of the assessments through ageing – using an ageing factor. We use simple aggregation by adding the scalar value corresponding to the current opinion update and to the latest overall trust value.We also propose a classifier that places any subjective opinion into one of six rating classes, which subsequently help to decide the scalar rating value corresponding to any provider or user opinion.The aggregation constant is a fixed value that is multiplied with the rating class value to achieve the update effect of the user assessment, which is necessary to apply it in the aggregation operation.In addition to the use of aggregation, the resulting trustworthiness value for a service can also be calculated using ageing between current and previous opinions, which aggregates the effect of the current opinion update with a ratio derived from the assessment history from the same user.The problem here is that the results are highly affected by the values of aggregation and ageing factors.

In this thesis, we also present experiments with the participation of real users to provide a good selection for the aggregation constant and for the ageing factor, resulting in minimal average error between the calculated update and the human opinion – whether for a user or for a tester. More specifically, an adaptive method for estimating the aggregation constant and ageing factor enables the user to assume different values for the parameters and then calculate the updates of various randomly generated opinions, asking testers for estimations based on opinion visualisations and, finally, using these values to provide a good estimation for the parameters.

Another contribution of the present research is to examine how to perform new user assessments by transferring trust and asking previous (trusted) users who have already completed assessments for their opinions regarding a service. We do this using transitivity as well as fusion operators.

A final contribution is the use of blockchain technology to remove untrusted users' assessments before updating the system. We present smart contracts that are based on a voting system, which decides in a decentralised way whether or not a user's assessment was done fairly; hence, this system decides whether the user's opinion, collected from the assessment process, should update the overall trust value or not. We provide pseudo code for the smart contracts to show how we control the voting policy among cloud users.

1.4.2 Publications

We have published two conference papers from this thesis work:

- Abdelmageed Algamdi, Frans Coenen, and Alexei Lisitsa. *Reputation system aggregation and ageing factor selection using subjective opinions classification. In Global Summit on Computer & Information Technology (GSCIT17)*

- Abdelmageed Algamdi, Frans Coenen, and Alexei Lisitsa. *A trust evaluation method based on the distributed cloud trust protocol (ctp) and opinion sharing. In International Conference on Computer Applications and Technology (ICCAT'17)*

1.4.3 Thesis Outline

This thesis is organised as follows:

- Chapter 2 presents background information on the CTP, subjective opinions, subjective logic and blockchain technology.

- Chapter 3 shows the suggested cloud infrastructure that enables both provider and user assessments. We will show the importance of each part of the infrastructure and demonstrate how this infrastructure helps cloud users in their trust queries.

- Chapter 4 demonstrates various ways of calculating trust based on provider and user assessments. It also introduces how each cloud user defines his or her own assessment weighting criteria and generates his or her opinion values using this criteria. In this chapter, related research on TR systems is also discussed.

- Chapter 5 outlines a voting system using blockchain technology that helps to secure the suggested trust reputation system from untrusted users' assessments by blocking their assessment updates. For each service, a group exists that contains all the users who did previous assessments for that service. Previous users can access the new users' answers to the questionnaire and review new user assessments. Then, former users vote by either accepting or rejecting the new user and his or her assessment. We apply a reward or penalty to each user based on his or her vote in comparison with the overall resulting vote. This chapter also presents related research work regarding the removal of unfair opinions.

- Chapter 6 examines the suggested TR system and shows the estimated values for the aggregation constant and ageing factor so that we can minimise the average error.
- Chapter 7 summarises the contributions of this thesis and suggests possibilities for future work.

Chapter 2

Background Information

2.1 Introduction

In this chapter, we provide background information regarding the CTP and how it allows users to obtain information. We also introduce CAIQs, which can be used by providers to conduct self-assessments about the services they offer. Moreover, we introduce subjective opinions (specifically, binomial subjective opinions), and some of the subjective logic operations that relate to our work. These operations comprise addition, multiplication, consensus and averaging operators. Finally, we discuss the fundamentals of blockchain technology and smart contracts that can be written with Ethereum.

2.2 The CTP

According to (38) and (39), the CTP is a protocol that enables a cloud service consumer to request and retrieve trust-related information from a cloud service provider. The information that may be received concerns the main attributes needed for the assessment of a service. These attributes are security, integrity, compliance, privacy and the operational security history of service elements. That is, the CTP enables the cloud user to ask for and obtain answers about the configuration of a service and other specifications of a service, as shown in (37); this can help a user complete an assessment of a cloud service provider and gain control regarding the provider, as mentioned in (37).

(65) noted that the main purpose of the CTP is to generate evidence that every aspect of a service is running, based on an SLA between a cloud user and a provider. The CTP is 'Transparency as a Service' (TaaS), and is often used to perform monitoring and to provide evidence-based assurance. The evidence is based on pieces of information

called EoTs. The elements offer proof regarding the important security configurations and functional characteristics of all systems that may potentially be integrated with a computing cloud. Additionally, they can be used to determine which cloud service is best suited to meeting processing requirements.

Moreover, as (50) explained, the CTP describes how a cloud consumer, or user, asks about an EoT and how a package may be provided to the user in response. The response is made through a request/response technique that involves 24 EoTs. The 24 EoTs represent all types of requests that the cloud user can ask make of a cloud service provider. The first two EoTs represent the initiation and the termination of any session related to the CTP. The other 22 EoTs represent information about specifications and controls. They are classified by type into the categories evidence requests, provider assertions, provider notifications, policy introductions, SCAPs and extensions. They are also classified by family into the categories configurations, vulnerabilities, anchors, audit logs, provider capabilities, client definitions, service statistics, service claims, service alerts, service users, service permissions, service configurations, service anchors, service quotas and service management. A complete list of EoTs is shown in Figure 9 in a paper by (37). With EoTs, the CTP is an adaptable protocol that can be adjusted according to the digital trust requirements of cloud consumers and the functional situations of cloud providers.

As aforementioned, the CTP's data model represents security, compliance and data governance attributes that can be asked for by the CTP's clients. According to (50), these attributes are represented by 10 structures. These structures are customers, assets, attributes, measurements, metrics, triggers, results, objectives, log entries and service views. These structures represent services offered, characterise elements of cloud systems (physical or ethereal) through assets, denote sets of security attribute measurements and describe standardisations of measurements in metrics, triggers and log entries that relate to request/response situations regarding measurements required by consumers, as mentioned in (50).

The CTP has a RESTful API that performs request/response queries using HTTPs, such as 'get', 'put', 'post' and 'delete'. Thus, the CTP does not require a specific infrastructure to work, as (51) noted. Table 2.1 has examples of HTTP response codes to a request given to the CTP. '200' is an HTTP response code that denotes 'Yes' or 'OK', while '404' is an HTTP response code that denotes 'No'. Similarly, '204' is an HTTP response code that acknowledges that a user's request arrived but a cloud provider does not want to respond, and '401' is an HTTP response code that indicates

Request	GET https://cloudtrust.csc.com/ctp/[custID] /resources//cpe/?tag=infra&start=1&end=2
Response	HTTP/1.1 200 Content Length: 942 Content Type: text/xml *Information about the CTP version*
Request	GET https://cloudtrust.csc.com/ctp/[custID] /resources/loc/60a76c80d39911d9b93C0003939e0af6
Response	HTTP/1.1 200 Content Length: 3 US *The server name*
Request	GET https://cloudtrust.csc.com/ctp/[custID] /resources/2/
Response	HTTP/1.1 200

Table 2.1: A CTP request/response process.

unauthorised requests were made. Table 2.1 also provides a complete example of how a cloud user may ask a provider whether data is stored in the provider's server or not. The user may begin by initiating the CTP request of the provider by asking for EOT number one, while the provider may accept the request with the response code '200'. The response message may contain information about the CTP version that is being used. Then, the user may request information about EOT number eight. If data is stored on the provider's server, the HTTP response code '200' may be sent with the name of the server and the length of the data on the server. However, if the data is not available, the HTTP response code '404' may be sent. The last request may be to terminate the CTP session with the provider by requesting EOT number two, which may be answered with the HTTP response code '200'.

2.3 Assessment Models

2.3.1 CAIQ Provider Models

According to (18), the Cloud Security Alliance (CSA) generated a spreadsheet containing 293 'Yes' and 'No' questions; the spreadsheet is known as the CAIQ, which covers the main attributes, such as compliance and data governance, used in each assessment process. As shown in Appendix B, the CAIQ has 133 controls in its framework. Each control has one question or more about various cloud providers' capabilities and competencies. It can be adopted to offer cloud customers a means to make requests of providers

without compromising infrastructure security. Additionally, the CAIQ can reduce burdens on cloud providers that are caused by numerous queries. Finally, the CAIQ also assists both cloud customers and cloud auditors when they evaluate cloud providers; see (18), (16) and (55).

In fact, to elaborate, according to (18), (16) and (55), the CAIQ involves 16 attributes, or domains; they are listed below.

- *Application and interface security.* These attributes contain information regarding applications that a provider's employees use.
- *Audit assurance and compliance.* These attributes contain information that confirms that an audit operation is sufficient to be applied to a cloud and determines whether a cloud service provider assessed his or her environment by reviewing the efficacy of his or her service's security operation implementations.
- *Business continuity management and operational resilience.* These attributes contain information regarding the operating ability of a service and the service's outage event priorities, such as testing and maintenance.
- *Change control and configuration management.* These attributes contain information about whether a cloud service provider tracks any changes on his or her cloud and whether, if any changes are made, the provider follows an internal system process.
- *Data security and information lifecycle management.* These attributes contain information about whether a cloud service provider knows the importance of his or her data and about what controls the provider uses to preserve this data.
- *Data centre security.* These attributes contain information about the physical controls a cloud service provider has.
- *Encryption and key management.* These attributes contain information about the encryption options a cloud service provider offers and whether or not the options are appropriate.
- *Governance and risk management.* These attributes contain information about whether a cloud service provider has a risk assessment program, particularly a program associated with data governance, and whether the cloud service provider publishes risk assessment reports that can be accessed by cloud users.

- *Human resources.* These attributes contain information regarding employees' behaviours, training and skills. The information can be used to assess privacy breaches by employees, for example, and whether users are notified when such breaches occur.

- *Identity and access management.* These attributes contain information regarding cloud service providers and federations. The information can be used to ensure that users' accounts are correctly managed by a provider through controlled log access.

- *Infrastructure and virtualisation security.* These attributes contain information about IaaS security, such as how audit logs stored, reviewed and accessed by service providers.

- *Interoperability and portability.* These attributes contain information regarding policies and documents noted in SLAs that enable cooperation between services and third-party applications.

- *Mobile security.* These attributes contain information regarding mobile security, such as anti-malware, compatibility, device management and encryption functions.

- *Security incident management, e-discoveries and cloud forensics.*These attributes contain information about incident responses and forensic capabilities regarding security incidents in cloud systems consumers use.

- *Supply chain management, transparency and accountability.* These attributes contain information about the management of contracts by appropriate resources and about the availability of incident details and reports to users.

- *Threat and vulnerability management.* These attributes contain information about whether cloud service providers have appropriate threat and vulnerability processes in regards to mitigation and protection.

(22), (18) and (9) used the CAIQ for self-assessments and to construct a reputation system based on opinions extracted from questionnaire answers. The reputation system used only the CAIQ opinions and was based on one-sided trust with which a provider submitted certificates about his or her service's operation. The system used only these certificates, and providers completed the self-assessments.

It should be noted that trust relationships are constructed over time, and a service's reputation should reflect community aggregated opinions about the service. Thus, our work enables cloud users (after spending time using services) to submit their opinions about service operations using an assessment questionnaire designed specifically for users; the questionnaire is shown in the next section. Users' opinions (i.e., community opinions) are aggregated and used to update the initial reputation of a service that is generated based on a provider's self-assessment.

2.3.2 Smals ICT for Society User Assessment Models

Smals ICT for Society—see (44)—released two assessment models that can be used by clients to assess cloud services. The first model can be used by experts, who may be asked about the requirement levels (governance, IAM, IT security and operational security levels) that are necessary for providing a service to users. (44) noted that the benefit of this assessment model is that it provides cloud providers information about users' needs that can be utilised before providers launch their services over clouds (this model is used during service design stages). However, this model is inappropriate for normal users who need to assess services. Thus, the other assessment model was designed for normal clients in order to allow them to assess cloud services based on their experiences using the services. The users are asked questions, most of which are multiple choice. Then, the assessment model studies the same four domains as the expert assessment model: governance, IAM, IT security and operational security.

According to see (44), governance questions often involve four sub-domains: legal implications, supply chain management, audits and business continuity. In contrast, IAM questions often involve authentication levels, user management and access management. Moreover, IT security questions often involve different aspects, such as data segregation, interface security, input/output data integrity, infrastructure security and virtualisation security, which investigate security approaches to wired/wireless networks and hardware and virtualisation levels. These questions also involve cryptography and encryption/decryption approaches. Finally, operational security questions often involve backup recovery, disaster recovery, incident management and vulnerability management.

2.4 Subjective Logic and Subjective Opinions

Subjective logic was introduced by (28). The concept was based on belief functions discussed by (57). These functions were proposed as a way to model epistemic uncertainty. According to (30), subjective logic can be seen as an extension of standard logic, as per (12), and probabilistic logic, as per (49).

Probabilistic logic considers uncertainty and belief ownership. It is suitable for considering models with uncertainty and incomplete knowledge (which are essential for the assessment method we use in this paper). Subjective logic, an extension of this, is the most suitable form of logic for addressing human emotions as no human has absolute emotions (e.g., no human is 100% certain) about anything. For example, suppose that person A, who owns a restaurant, believes that he or she has a 70% chance to double his or her restaurant's profit if he or she invests a significant amount of money in marketing his or her business. It can be concluded that this person has a 70% belief that he or she can increase his or her restaurant's profit in this manner, while this person has a 30% disbelief or uncertainty that he or she will not increase his or her restaurant's profits.

According to (31; 35), the advantage of using subjective logic is the ability to distinguish between certain and uncertain conclusions as uncertainty is considered in subjective logic calculations. Moreover, subjective logic uses operations that work with subjective opinions (specifically, in regards to this study, binomial subjective opinions). These operations include addition, subtraction, multiplication, comultiplication and complementary functions.

For the purpose of this study, we were only interested in multiplication and opinion consensus operators. We used independent consensus operators to gather the opinions of two different agents, or users, about a single predicate, or service, and combine them to reflect the users' overall opinion of the service. However, we used multiplication operations instead of consensus operations for some situations because consensus operations could not be used for opinions that are certain. Additionally, we used dependent consensus operations, also known as averaging operations, to combine different users' dependent opinions about a single service.

2.4.1 Opinion Representation

Feedback from cloud service consumers can be modelled as subjective opinions. A normal subjective opinion can be represented by a tuple comprising the main values belief mass, uncertainty mass and base rate. Belief mass and uncertainty mass distributions over a

domain, $\mathbb{X}$, or a hyperdomain, $\rho(\mathbb{X})$, are defined as b_x and u_x, respectively, where x is a variable in the domains. A domain, $\mathbb{X}$, is a state space representation of n values, which can equal two (true or false, in binary code) or greater than two. A hyperdomain, $\rho(\mathbb{X})$, is a reduced powerset of a domain, or $\mathbb{X}$. b_x represents a belief about the element x that belongs to the domain $\mathbf{X}$, while u_x represents ignorance or a lack of information (i.e., an unknown belief). Belief and uncertainty mass distributions complement each other, which means that the sum of two mass distributions is one, as shown in 2.1:

$$u_x + \sum_{x \in \mathbb{X}} b_x = 1 \text{ where } b_x{:}\mathbb{X} \rightarrow [0, 1]. \tag{2.1}$$

b_x represents a mass over possible values that belong to domain X or hyperdomain $\rho(\mathbb{X})$, while u_x represents a mass that does not have any possible value.

According to (35) and (31), the base rate distribution a_x is calculated using probability theory. The default base rate of a singleton that belongs to domain $\mathbb{X}$ of cardinality m equals $1/m$. The sum of all base rate distributions over all possible values, x, that belong to domain $\mathbb{X}$ is 1, is shown in equation 2.2:

$$\sum_{x \in \mathbb{X}} a_x = 1 \text{ where } a_x{:}\mathbb{X} \rightarrow [0, 1]. \tag{2.2}$$

As explained by (35) and (31), a subjective opinion, denoted by subject agent A, regarding the specific target variable x, is denoted by $\omega_x^A = (b_x, u_x, a_x)$. In our system, each subject agent represents a cloud user, while each target variable represents a service. ω_x^A represents a user's belief about a target variable (i.e., a service operation). Each operation can be converted into a trust relationship between an agent (i.e., a cloud consumer, or user) and a target variable owner (i.e., a service provider) after an assessment is completed.

Moreover, according to (35) and (31), subjective opinions can be divided into 12 classes. Classes are divided based on two main properties. The first involves a domain and its variables, and the second involves a level of uncertainty and a belief mass value. Moreover, based on domain classifications, opinions can be classified as binomial opinions, multinomial opinions and hyperopinions. True and false representations with cardinalities of 2 are used in binomial opinions, while such representations with cardinalities of greater than 2 are used in multinomial opinions. Such representations with cardinalities of greater than 2 are also used in hyperopinions. However, hyperopinions work in regards to hyperdomains. Each hyperopinion is divided based on the values of a belief mass, b_x, and an uncertainty mass, u_x, into one of the following classes: vacuous, or

($u_x = 1$); uncertain, or ($0 < u_x < 1$) ; dogmatic, or ($u_x = 0$) and absolute, or ($b_x = 1$). Our study's proposed assessment operations use a 'Yes' and 'No' questionnaire. The most appropriate subjective opinion class for this type of assessment is binomial.

Formal Definitions

Consider the binary domain $\mathbb{X} = \{TRUE, FALSE\}$. Let x be a random binomial variable in the domain $\mathbb{X}$. A binomial opinion about a truth state, x, can be described as a tuple consisting of four values, $\omega_x = (b_x, d_x, u_x, a_x)$, such that $b_x + d_x + u_x = 1$. We have two variables to express a belief mass distribution rate: b_x, for a belief mass with which $x = TRUE$, and d_x, for a disbelief mass with which $x = FALSE$. Moreover, u_x is an uncertainty mass for an uncommitted belief/disbelief mass (i.e., $x \notin \mathbb{X}$ and $x =$ unknown answer). Finally, a base rate is expressed by a_x, as described by (31; 35).

Barycentric Coordinates

A binomial subjective opinion $\omega_x^A = (b_x, d_x, u_x, a_x)$ can be visualised using barycentric coordinates, as per (23) and (31), inside a triangle with uncertainty, belief and disbelief vertices, as shown in Figure 2.1. The triangle in the figure has equal sides. An opinion is represented as a centre of gravity (a barycentre or a geometric centroid) for locating three masses (i.e., M_A, M_B and M_C), at the triangle's vertices. These masses are located over three perpendicular axes opposite the vertices in the triangle. These masses are represented by b_x, d_x and u_x, respectively. A base rate, a_x, is represented by a point in the base. A line connecting an uncertainty vertex to a point represented by a_x is called a director. The projected probability, P_x, of an opinion, w_x, can be determined by drawing a line from an opinion point, ω_x, to a base that is parallel to a director.

For homogeneous barycentric coordinates, edges are normalised in order to achieve $b_x + d_x + u_x = 1$. A projected probability can be calculated as follows: $P_x = b_x + u_x a_x$. Subjective opinions can be categorised into one of the following subclasses.

- *Absolute belief/disbelief binomial.* $b_x = 1$ or $d_x = 1$, respectively; an absolute opinion with total belief is shown using ω_{x_1} in Figure 2.2.
- *A dogmatic binomial.* $u_x = 0$; a dogmatic opinion is shown using ω_{x_2} in Figure 2.2.
- *A vacuous opinion.* $u_x = 1$; a vacuous opinion is shown using ω_{x_3} in Figure 2.2.

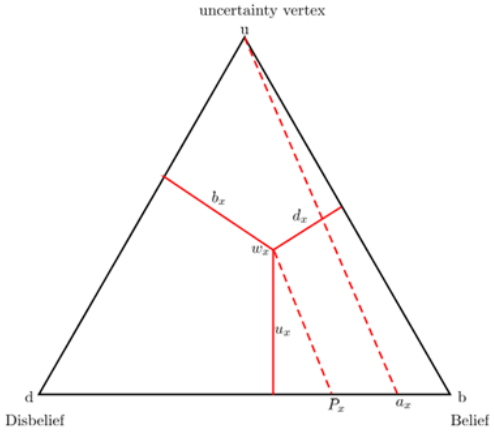

Figure 2.1: A binomial opinion representation inside barycentric coordinates; see (31).

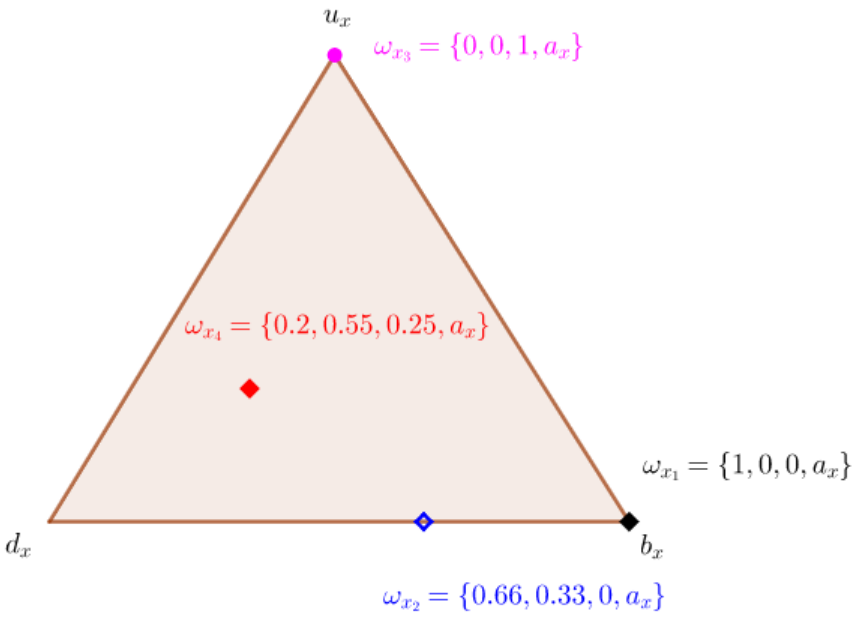

Figure 2.2: Different types of binomial opinions.

- *An uncertain opinion.* $0 < u_x < 1$; an uncertain opinion is shown using ω_{x_4} in Figure 2.2.

A binomial opinion has a projected probability of $P_x = b_x + a_x u_x$ and a variance of $Var_x = \frac{P_x(1-P_x)u_x}{W+u_x}$, where W is a non-informative weight that is set to 2 in the opinion class (i.e., binomial).

2.4.2 Subjective Logic

For the two binomial opinions ω_{x_1} and ω_{x_2} in regards to the random binomial variables x_1 and x_2, respectively,

$$\omega_{x_1} = (b_{x_1}, d_{x_1}, u_{x_1}, a_{x_1})$$

and

$$\omega_{x_2} = (b_{x_2}, d_{x_2}, u_{x_2}, a_{x_2})$$

.

Opinion Addition

According to (31), the addition (union) of two opinions, ω_{x_1} and ω_{x_2}, is denoted by $\omega_{(x_1 \cup x_2)} = (b_{(x_1 \cup x_2)}, d_{(x_1 \cup x_2)}, u_{(x_1 \cup x_2)}, a_{(x_1 \cup x_2)})$, which conditions $x_1 \cup x_2 \subset \mathbb{X}$. This means that together, x_1 and x_2 do not represent a complete partition of $\mathbb{X}$. Note that addition is a binary operation that involves two binomial opinions, ω_{x_1} and ω_{x_2}, and the operation can produce an output opinion that represents a union of the two input opinions, as per (31):

$$\begin{aligned} b_{(x_1 \cup x_2)} &= b_{x_1} + b_{x_2} \\ d_{(x_1 \cup x_2)} &= \frac{a_{x_1}(d_{x_1} - b_{x_2}) + a_{x_2}(d_{x_2} - b_{x_1})}{a_{x_1} + a_{x_2}} \\ u_{(x_1 \cup x_2)} &= \frac{a_{x_1} u_{x_1} + a_{x_2} u_{x_2}}{a_{x_1} + a_{x_2}} \\ a_{(x_1 \cup x_2)} &= a_{x_1} + a_{x_2} \end{aligned}$$

.

The above has a projected probability of $P_{x_1 \cup x_2} = P_{x_1} + P_{x_2}$; see (35) and (31). An example of a union of two binomial opinions is shown in Figure 2.3. The first opinion is $\omega_{x_1} = \{0.25, 0.35, 0.4, 0.75\}$, with a projected probability of $P_{x_1} = 0.55$. The second

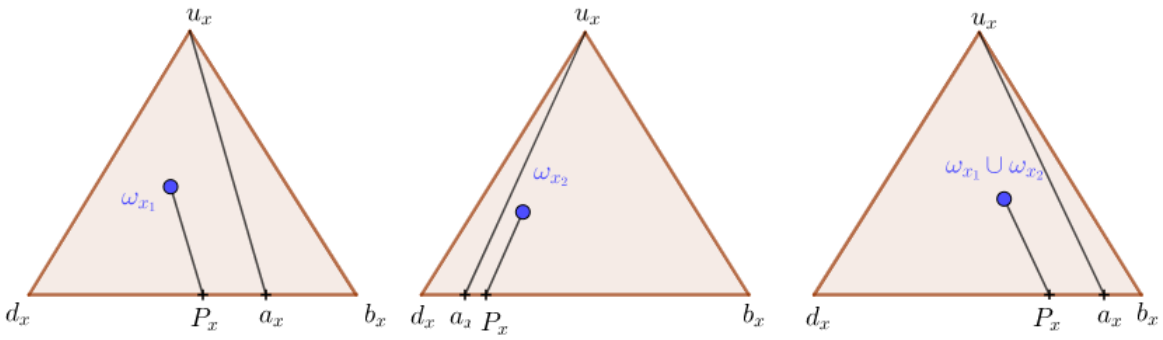

Figure 2.3: An addition of two binomial opinions.

opinion is $\omega_{x_2} = \{0.15, 0.55, 0.3, 0.15\}$, with a projected probability of $P_{x_2} = 0.195$. The resulting opinion after addition is $\omega_{x_1 \cup x_2} = \{0.4, 0.2167, 0.3833, 0.9\}$, with a projected probability of $P_{x_1 \cup x_2} = 0.74497$. The addition of the two binomial opinions generated an overall opinion with a belief higher than both of the input opinions (i.e., the overall opinion equalled the addition of the two beliefs).

Opinion Multiplication

The multiplication (the logic 'AND' operator) of opinions w_{x_1} and w_{x_2} is denoted by $\omega_{(x_1 \wedge x_2)} = (b_{(x_1 \wedge x_2)}, d_{(x_1 \wedge x_2)}, u_{(x_1 \wedge x_2)}, a_{(x_1 \wedge x_2)})$:

$$b_{(x_1 \wedge x_2)} = b_{x_1} b_{x_2} + \frac{(1 - a_{x_1}) a_{x_2} b_{x_1} u_{x_2} + a_{x_1} (1 - a_{x_2}) u_{x_1} b_{x_2}}{1 - a_{x_1} a_{x_2}}$$

,

$$d_{(x_1 \wedge x_2)} = d_{x_1} + d_{x_2} - d_{x_1} d_{x_2}$$

,

$$u_{(x_1 \wedge x_2)} = u_{x_1} u_{x_2} + \frac{(1 - a_{x_2}) b_{x_1} u_{x_2} + (1 - a_{x_1}) u_{x_1} b_{x_2}}{1 - a_{x_1} a_{x_2}}$$

and

$$a_{(x_1 \wedge x_2)} = a_{x_1} a_{x_2}$$

.

The projected probability is $P_{(x_1 \wedge x_2)} = P_{x_1} P_{x_2}$; see (35) and (31). According to the authors, binomial multiplication is a binary operation with two distinct binary domains, or binomial opinions, as inputs and one domain, or binomial opinion, as an output.

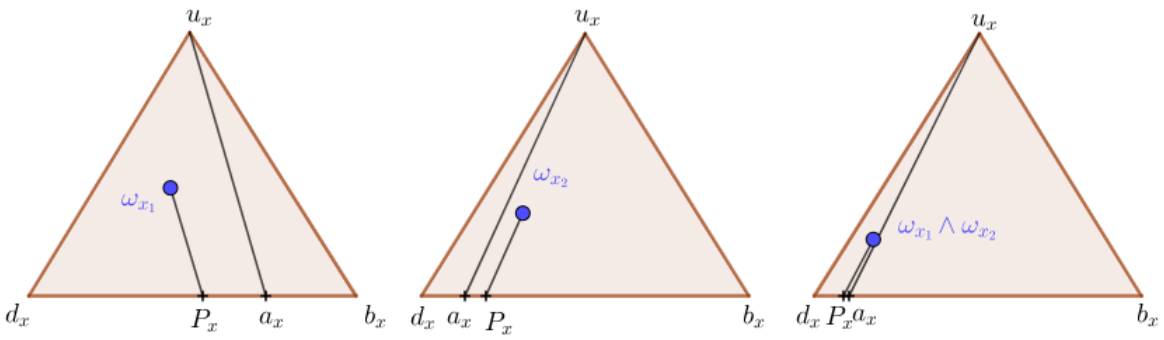

Figure 2.4: A multiplication of two binomial opinions.

The output relates to a subset of the inputs in regards to a Cartesian product, where the binary domains' variables for the input opinions exist in true forms. For example, suppose that the domains for input binomial opinions are $\mathbb{X} = \{x, \overline{x}\}$ and $\mathbb{Y} = \{y, \overline{y}\}$. The domain for the Cartesian product is $\mathbb{X} \times \mathbb{Y} = \{(\overline{x}, \overline{y}), (x, \overline{y}), (\overline{x}, y), (x, y)\}$, and the domain for the binomial product is $\{(x, y)\}$.

An example of the multiplication of two binomial opinions is shown in Figure 2.4. The first opinion is $\omega_{x_1} = \{0.25, 0.35, 0.4, 0.75\}$, with a projected probability of $P_{x_1} = 0.55$. The second opinion is $\omega_{x_2} = \{0.15, 0.55, 0.3, 0.15\}$, with a projected probability of $P_{x_2} = 0.195$. The resulting opinion after multiplication is $\omega_{x_1 \wedge x_2} = \{0.0838, 0.7075, 0.2087, 0.1125\}$, with a projected probability of $P_{x_1 \wedge x_2} = 0.1073$. The multiplication of the two binomial opinions generated an overall opinion with a disbelief higher than both of the input opinions.

Independent Opinion Consensus

Suppose we have two independent opinions from different agents, or users, about a certain service. A consensus opinion generated from both the independent opinions reflects both the opinions in a fair and equal manner. Specifically, suppose the users, A and B, have the opinions $w_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$ and $w_x^B = (b_x^B, d_x^B, u_x^B, a_x^B)$, respectively, about service x. The consensus opinion is denoted by $w_x^{A \diamond B} = w_x^A \oplus w_x^B = (b_x^{A \diamond B}, d_x^{A \diamond B}, u_x^{A \diamond B}, a_x^{A \diamond B})$. This can apply to two cases. For the first case,

$u_x^A + u_x^B - u_x^A u_x^B \neq 0$,

$$b_x^{A \diamond B} = \frac{b_x^A u_x^B + b_x^B u_x^A}{\kappa}$$

,

$$d_x^{A\diamond B} = \frac{d_x^A u_x^B + d_x^B u_x^A}{\kappa}$$

,

$$u_x^{A\diamond B} = \frac{u_x^A u_x^B}{\kappa}$$

and

$$a_x^{A\diamond B} = \frac{a_x^B u_x^A + a_x^A u_x^B - (a_x^A + a_x^B) u_x^A u_x^B}{u_x^A + u_x^B - 2u_x^A u_x^B}$$

,

where $\kappa = u_x^A + u_x^B - u_x^A u_x^B$. Note that this operation cannot be applied to vacuous (i.e., $u_x = 1$) or dogmatic (i.e., $u_x = 0$) opinions. It is conditioned so that it can be applied to only uncertain opinions, $(0 < u_x < 1)$; see (34; 69).

For the second case,

$u_x^A + u_x^B - u_x^A u_x^B = 0$,

$$b_x^{A\diamond B} = \frac{\gamma^{A/B} b_x^A + b_x^B}{\gamma^{A/B} + 1}$$

,

$$d_x^{A\diamond B} = \frac{\gamma^{A/B} d_x^A + d_x^B}{\gamma^{A/B} + 1}$$

,

$$u_x^{A\diamond B} = 0$$

and

$$a_x^{A\diamond B} = \frac{\gamma^{A/B} a_x^A + a_x^B}{\gamma^{A/B} + 1}$$

.

According to see (34; 69), this is a case in which there are certain opinions. Their relative weight is $\gamma^{A/B} = lim(\frac{u_x^B}{u_x^A})$. Note that the main objective of a consensus operator is to obtain evidence for a resulting opinion using two combined input opinions.

An example of a consensus between two binomial opinions is shown in Figure 2.5. The first opinion is $\omega_x^A = \{0.25, 0.35, 0.4, 0.75\}$, with a projected probability of 0.55. The second opinion is $\omega_x^B = \{0.15, 0.55, 0.3, 0.15\}$, with a projected probability of 0.195. The resulting opinion after a consensus is $\omega_x^{A\diamond B} = \{0.233, 0.56, 0.207, 0.385\}$, with a projected probability of 0.313.

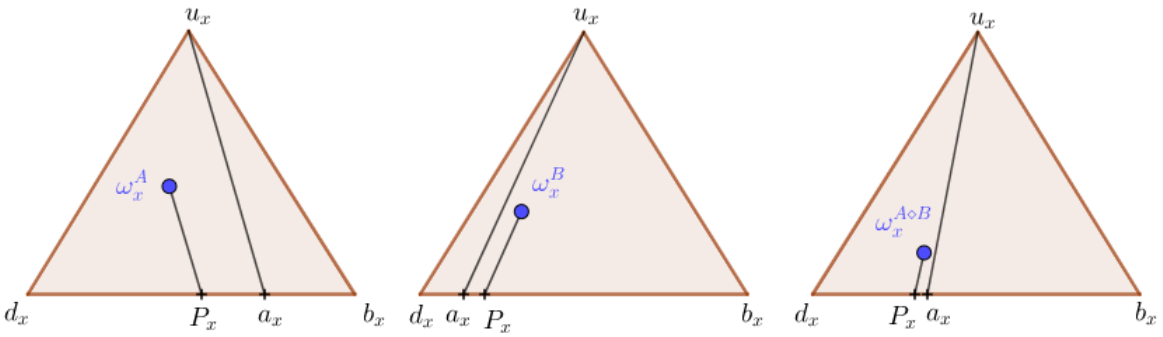

Figure 2.5: A consensus of two binomial opinions.

Dependent Opinion Averaging

According to (34) and (69), for a set of dependent opinions, a consensus operator is referred to as an averaging operator; it is different from consensus operators, which address independent opinions. Dependent operators share an information source, and duplicating the same information many times can result in errors. $\underline{\oplus}$ denotes an average fusion of binomial opinions.

Let a set of dependent opinions be denoted by $\omega_x^{A_i} = (b_x^{A_i}, d_x^{A_i}, u_x^{A_i}, a_x^{A_i})$, where the opinion $i \in [1, n]$ is held by agents $A_1, ..., A_n$ regarding proposition (i.e., service) x. Then, let an averaging operation for these dependent opinions be denoted by $\omega_x^{A_1} \underline{\diamond} ... \underline{\diamond} \omega_x^{A_n} = (b_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n}, d_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n}, u_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n}, a_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n})$, where

$$b_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n} = \frac{\sum_1^n (b_x^{A_i} / u_x^{A_i})}{\sum_1^n (b_x^{A_i} / u_x^{A_i}) + \sum_1^n (d_x^{A_i} / u_x^{A_i}) + n}$$

,

$$d_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n} = \frac{\sum_1^n (d_x^{A_i} / u_x^{A_i})}{\sum_1^n (b_x^{A_i} / u_x^{A_i}) + \sum_1^n (d_x^{A_i} / u_x^{A_i}) + n}$$

,

$$u_x^{A_1 \underline{\diamond} ... \underline{\diamond} A_n} = \frac{n}{\sum_1^n (b_x^{A_i} / u_x^{A_i}) + \sum_1^n (d_x^{A_i} / u_x^{A_i}) + n}$$

and

$$a_x^{A_1 \underline{\diamond} \ldots \underline{\diamond} A_n} = \frac{\sum_1^n a_x^{A_i}}{n}$$

.

The equations above can be applied when there is at least one opinion where $u_x^{A_i} \neq 0, i \in [1,n]$. However, if there is an opinion with zero uncertainty (i.e., if there is a certain opinion), these equations can produce undetermined values. In the following example, these equations are converted to forms that can be used with certain opinions as well:

$$\begin{aligned}
b_x^{A_1 \underline{\diamond} \ldots \underline{\diamond} A_n} &= \frac{\sum_1^n (b_x^{A_i}/u_x^{A_i})}{\sum_1^n (b_x^{A_i}/u_x^{A_i}) + \sum_1^n (d_x^{A_i}/u_x^{A_i}) + n} \\
&= \frac{(b_1/u_1 + \ldots + b_n/u_n)}{(b_1/u_1 + \ldots + b_n/u_n) + (d_1/u_1 + \ldots + d_n/u_n) + n} \\
&= \frac{(\sum_{i\in[1,n]} b_x^{A_i} \prod_{j\in[1,n]\setminus\{i\}} u_x^{A_j}) / \prod_{i\in[1,n]} u_x^{A_i}}{\sum_{i\in[1,n]} \frac{1-u_x^{A_i}}{u_x^{A_i}} + n} \\
&= \frac{(\sum_{i\in[1,n]} b_x^{A_i} \prod_{j\in[1,n]\setminus\{i\}} u_x^{A_j}) / \prod_{i\in[1,n]} u_x^{A_i}}{\sum_{i\in[1,n]} 1/u_x^{A_i}} \\
&= \frac{\sum_{i\in[1,n]} (b_x^{A_i} \prod_{j\in[1,n]\setminus\{i\}} u_x^{A_j})}{\sum_{i\in[1,n]} \prod_{j\in[1,n]\setminus\{i\}} u_x^{A_j}}.
\end{aligned} \tag{2.3}$$

Similarly,

$$\begin{aligned}
d_x^{A_1 \underline{\diamond} \ldots \underline{\diamond} A_n} &= \frac{\sum_1^n (d_x^{A_i}/u_x^{A_i})}{\sum_1^n (b_x^{A_i}/u_x^{A_i}) + \sum_1^n (d_x^{A_i}/u_x^{A_i}) + n} \\
&= \frac{\sum_{i\in[1,n]} (d_x^{A_i} \prod_{j\in[1,n]\setminus\{i\}} u_x^{A_j})}{\sum_{i\in[1,n]} \prod_{j\in[1,n]\setminus\{i\}} u_x^{A_j}},
\end{aligned} \tag{2.4}$$

$$
\begin{aligned}
u_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} &= \frac{n}{\sum_1^n (b_x^{A_i}/u_x^{A_i}) + \sum_1^n (d_x^{A_i}/u_x^{A_i}) + n} \\
&= \frac{(n \prod_{i \in [1,n]} u_x^{A_i})}{\sum_{i \in [1,n]} \prod_{j \in [1,n] \setminus \{i\}} u_x^{A_j}} \\
&= 1 - (b_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} + d_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n})
\end{aligned}
\tag{2.5}
$$

and

$$
a_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = \frac{\sum_{i=1}^n a_x^{A_i}}{n}. \tag{2.6}
$$

We can use the equations above to average n binomial opinions, in which there is at least one opinion with non-zero uncertainty. However, consider an example in which there are $(n-1)$ binomial opinions with uncertain values, (i.e., $u_x^{A_i} \neq 0, i \in [1, n-1]$), and there is only one opinion with certainty (i.e., $u_x^{A_i} = 0, i = n$). The resulting opinion after averaging is $u_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = 0$, which is unfair as the non-zero uncertainty values of $n-1$ opinions were not considered in the operations.

A solution to this situation is provided below. The resulting average binomial opinion of n dependent opinions, $\omega_x^{A_1} \underline{\diamond} \ldots \underline{\diamond} \omega_x^{A_n} = (b_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n}, d_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n}, u_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n}, a_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n})$, can be calculated as follows.

If $\forall i \in [1, n], u_x^{A_i} \neq 0$, then

$$
\begin{aligned}
b_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} &= \frac{\sum_1^n (b_x^{A_i}/u_x^{A_i})}{\sum_1^n (b_x^{A_i}/u_x^{A_i}) + \sum_1^n (d_x^{A_i}/u_x^{A_i}) + n} \\
&= \frac{\sum_{i \in [1,n]} (b_x^{A_i} \prod_{j \in [1,n] \setminus \{i\}} u_x^{A_j})}{\sum_{i \in [1,n]} \prod_{j \in [1,n] \setminus \{i\}} u_x^{A_j}},
\end{aligned}
\tag{2.7}
$$

$$
\begin{aligned}
d_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} &= \frac{\sum\limits_{1}^{n}(d_x^{A_i}/u_x^{A_i})}{\sum\limits_{1}^{n}(b_x^{A_i}/u_x^{A_i}) + \sum\limits_{1}^{n}(d_x^{A_i}/u_x^{A_i}) + n} \\
&= \frac{\sum\limits_{i\in[1,n]}(d_x^{A_i} \prod\limits_{j\in[1,n]\setminus\{i\}} u_x^{A_j})}{\sum\limits_{i\in[1,n]} \prod\limits_{j\in[1,n]\setminus\{i\}} u_x^{A_j}},
\end{aligned}
\tag{2.8}
$$

$$
\begin{aligned}
u_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} &= \frac{n}{\sum\limits_{1}^{n}(b_x^{A_i}/u_x^{A_i}) + \sum\limits_{1}^{n}(d_x^{A_i}/u_x^{A_i}) + n} \\
&= \frac{(n \prod\limits_{i\in[1,n]} u_x^{A_i})}{\sum\limits_{i\in[1,n]} \prod\limits_{j\in[1,n]\setminus\{i\}} u_x^{A_j}} \\
&= 1 - (b_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} + d_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n})
\end{aligned}
\tag{2.9}
$$

and

$$
a_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = \frac{\sum\limits_{i=1}^{n} a_x^{A_i}}{n}. \tag{2.10}
$$

However, if $\exists i \in [1, n], u_x^{A_i} = 0$, then

$$
b_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = \gamma \sum_{i\in[1,n]} b_x^{A_i}, \tag{2.11}
$$

$$
d_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = \gamma \sum_{i\in[1,n]} d_x^{A_i}, \tag{2.12}
$$

$$
d_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = \gamma \sum_{i\in[1,n]} d_x^{A_i} \tag{2.13}
$$

and

$$a_x^{A_1 \underline{\diamond} \cdots \underline{\diamond} A_n} = \gamma \sum_{i \in [1,n]} a_x^{A_i}, \tag{2.14}$$

where $\gamma = 1/n$.

2.5 Ethereum and Blockchain Technology

Ethereum is an open source platform that was proposed by (13). It was built based on blockchain technology as per (52) in order to create the ability to decentralise applications using smart contracts. (11) wrote scripting languages, such as Solidity; see (19). With a blockchain, complex computations can be performed where nodes, known as miners, can share computing resources. A blockchain is a chain of transactional records, and when a new transaction occurs, miners compete to compute the next block in the blockchain in order to earn a reward.

According to (19), Ethereum has cryptocurrency tokens, known as ethers. Ethers can be transferred between accountsdirectly, through transactions, or indirectly, through computation operations. The tokens can be seen as computational fees with which a miner pays for a requested computation. Moreover, ethers can be used to reward miners that help with computations by sharing computational resources; this help is often referred to as mining.

2.5.1 Trust, Privacy and Security in Blockchains

Blockchains are based on a decentralised topology in which no single miner is responsible for making decisions. Transactions do not need third parties for verification as all miners compete to verify transactions. Thus, blockchains are often referred to as trust free blockchains.

According to (19), blockchains use public/private key cryptography to secure ether transactions. Moreover, with blockchains, an account has two keys: private and public. A private key has a password that should be kept secret by an ether account owner so that no one other than the owner knows it, while a public key is shared with others. Public keys are cryptographically generated addresses stored in a blockchain. If an old owner wants to send ether to a new owner, the new owner, who has a public key, sends his or her key to the old owner. Then, an ether transaction is initiated by the old owner with the new owner's public key, and the old owner, who has a private key, completes the

transaction with the private key. Any miner on the blockchain can see the transaction with the aid of a public key, but only the new owner can control the ether involved in the transaction. Moreover, according to (63), for security, the blockchain uses a hash function, SHA256, for an encryption. The function transforms input information into a long hash (i.e., a long string of output information) with a 256 bit length.

2.5.2 Public and Private Blockchains

Public Blockchains

According to (19), any miner that has internet access can join and access a blockchain without permission. These blockchains are called low trust blockchains.

Private Blockchains

(19) noted that miners need permission to join private blockchains, and only miners with approved requests can be included on a private blockchain's list of accessible miners. This list is known as a white list. Approval needed to join a blockchain is not given by single miners. Instead, approved miners vote in order to decide whether permission should be granted to an applying miner. These blockchains are called highly trusted blockchains.

2.5.3 Smart Contracts

According to (19) and (11), a smart contract is an intelligent block of code that runs over a blockchain written using C, Java and Solidity, which are programming languages commonly used to write smart contracts. Such contracts comprise some storage files, some program codes that control functions and some account balances (currencies are stored in accounts). Any user can create a smart contract and make it available by conducting an ether transaction on a blockchain. Users encode such contracts into blockchains using Ethereum to gain self-execution without interference from anyone. Once a contract is created, it cannot be altered; other blockchain users can see and verify it, and if someone tries to alter it, blockchain users detect that action and prevent it from occurring.

An explanation of this, as per (19) and (11), is shown in Figure 2.6. Contracts can send and receive currencies to other contracts or users depending on currency balances. In order to execute a smart contract's code, a gas (a fraction of ether) is paid by a contact owner. This payment is non-refundable and is generally sufficient for executing a

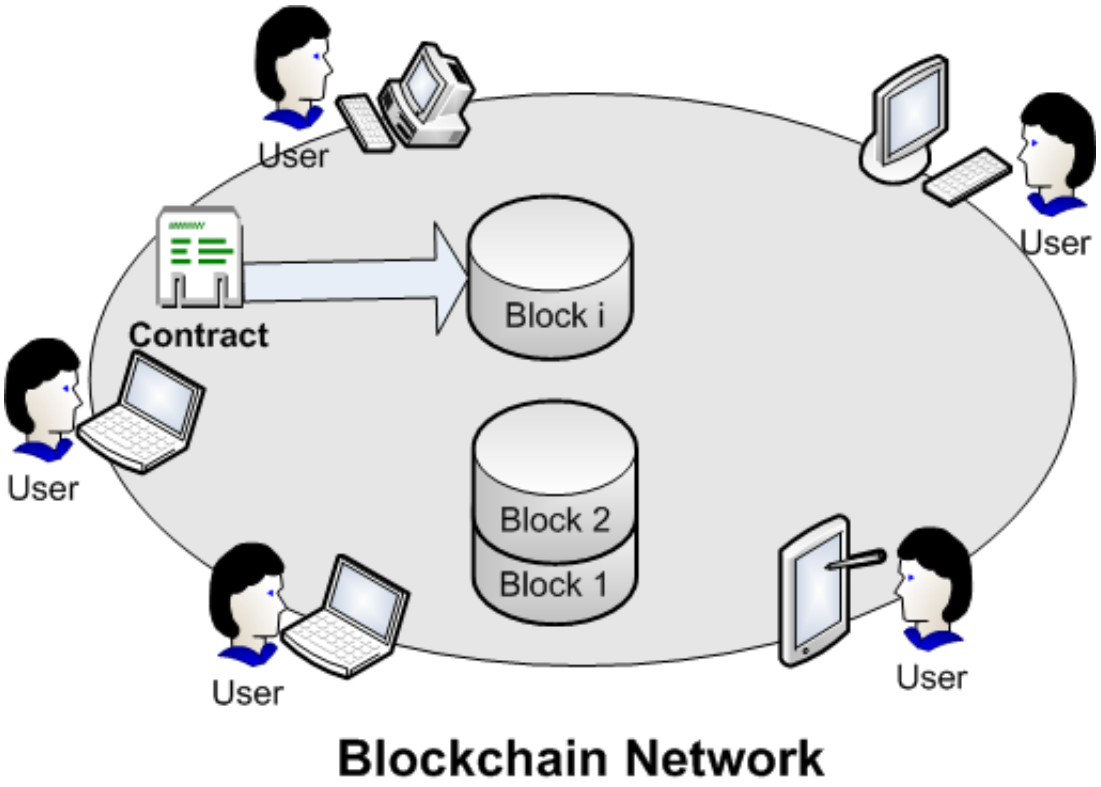

Figure 2.6: An example of a blockchain network.

contract code. Therefore, if the amount of paid gas is insufficient for executing the entire contract, the contract is not executed and the gas is not be refunded. More information about smart contracts, and examples of smart contracts written using Solidity, are in Section 5.3.

Chapter 3

Cloud Infrastructure for Trust Assessments

3.1 Introduction

In this chapter, we discuss a simple infrastructure that facilitates user assessments, provider assessments and the CTP We also propose an infrastructure for our trust system that is based on two different trust servers. The first is a general trust server that is public and can be seen by any node on a network. The second is a local trust server for each sub-network of users. Finally, we show how these servers act as routers to deliver different requests and responses.

3.2 The Proposed Infrastructure

Figure 3.1 shows the infrastructure we propose for our trust system. Any user might be a member of an organisation (i.e., a group). For every group of users of a cloud service, there is a local trust server (LTS), called an LTS/RqMg. An LTS/RqMg can calculate the trustworthiness of a specific provider and act as a request manager as part of the CTP. For every cloud service provider, a receiving manager (RsMg) is used to respond to a request sent by an RqMg through the CTP. A general trust server (GTS) is used to collect and store overall trust values about a cloud service provider gathered using an assessment operation. The GTS routes requests/responses according to the CTP. Finally, in an assessment operation, a CAIQ engine is used to store CAIQ answers from users.

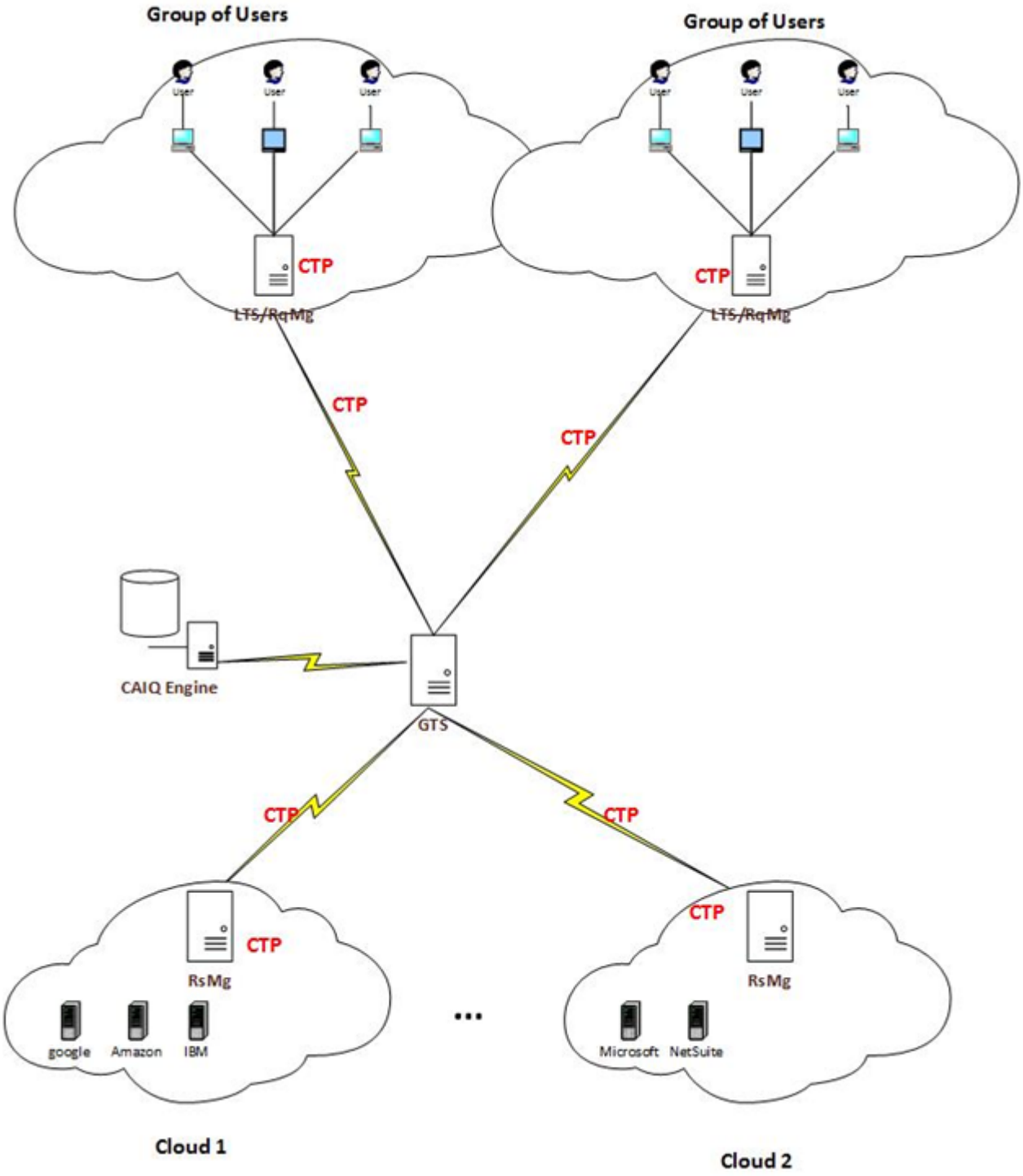

Figure 3.1: The proposed system infrastructure.

The CTP is used over a cloud by users to obtain information about a service and its operational attributes using HTTP requests/responses. The requests can be made to acquire information regarding 24 EoTs with the main attributes security, integrity, compliance, privacy and operational security. Requests/responses are made with the CTP using HTTP 'get' and 'put' methods, respectively. In a request with the CTP, the main data that sent is a provider's address as well as an EoT number, which a cloud user requests. In a response with the CTP, the main data that sent is an HTTP code that indicates whether an EoT request can be satisfied, as well as necessary data to prove that the request is or is not satisfied.

The infrastructure described in this chapter is designed to use the CTP in order to obtain trust information about any service provider. With this infrastructure, an assessment can be conducted in order to obtain a digital trust value for a provider. The system works in three modes: CTP, CAIQ and trust-retrieving requests, which are requests to an LTS and a GTS to give a user a list of all providers that offer a specific service based on trust values. As shown in a figure, a user can be placed into one of two group structures. A single user can build his or her own group and use a GTS directly; as the group comprises only one user, an LTS does not add any features to the group. In contrast, multiple users can build a group in which an LTS is used for trust references and a GTS is used as a cloud. The LTS is used only within the group of users. It is utilised to locally answer user trust requests depending on assessments from other users inside the group; these assessments are stored inside the LTS. However, if the data stored in the LTS is insufficient for answering a trust request from a user (e.g., if the user requests trust information about a new—that is, unassessed—service or data is out of date), the GTS is used for reference by the LTS. The GTS is also used to directly answer trust requests from normal single users that are not inside the group. The GTS acts as an accumulator of all the LTS's trust data.

3.2.1 LTS

As aforementioned, an LTS contains trust data about what has been assessed by members of its group. It is initially empty. Once a user assesses a specific provider, an entry is added that contains trust information about the specific provider's service, and a copy of the trust information is sent to a GTS. The trust data inside the LTS is organised as shown in Table 3.1, and each entry corresponds to a service offered by a cloud provider. Specifically, the trust is stored in two forms. The first form is used to store the trust's scalar value corresponding to the service's reputation, while the second form is used to

Domain URL (Name)	Service Type	Trust Value	Boolean Trust	Decision Time
...	...	...	...	...

Table 3.1: Trust data inside an LTS.

Service Type	Threshold Trust Value
...	...

Table 3.2: Service thresholds.

store the Boolean (i.e., true or false) value that indicates whether the service is trustworthy. If the service's scalar value passes a threshold trust value, shown in Table 3.2, it is trustworthy; otherwise, it is not. Finally, the time when the user's assessment was completed is stored.

The Trust Value column in an LTS contains a trustworthy value that has been calculated for an offered service using subjective logic operations, according to (22), (35) and (31). Similarly, the service's details are stored in the Domain URL and Service Type columns. Then, based on a threshold trust value in Table 3.2, regarding each service type, a true/false trust value is calculated and stored in the Boolean Trust column. Finally, the decision time is recorded and stored in the Decision Time column; this information is used in order to update the table.

3.2.2 GTS

A GTS contains all trust data gathered from all LTS servers and single users' assessments. The CTP requires 24 bits to identify its 24 EoTs. Moreover, the CTP requires providers' domain names and service names (codes).

3.2.3 The Importance of Using an LTS with a GTS

Consider a situation in which there are two providers, X and Y. They offer the same service and are trustworthy. Assume that due to an unfavourable distribution of provider X's servers, there is a place, L, where the provider's service does not work correctly. In fact, the problem provider X is experiencing is due to an incompatibility between the provider's hardware network structure and specific provider demands.

While both providers X and Y generally offer a good service, the users in L do not think that provider X does so. Moreover, as provider X's service does not work correctly

in L, it is not a good decision for users to ask provider X for the service. It is better for users to ask provider Y for the service instead.

In this situation, an LTS is very important as it relates to assessments based on typology. The LTS contains a list of all cloud service providers that offer the service that provider X offers that have been assessed and rated as trustworthy by users in the LTS's group. The LTS thus solves the physical infrastructure problem in regards to the cloud service provider servers, and the users are not considered in an assessment operation.

3.3 How the Proposed Infrastructure Works

Details regarding different requests/responses that can be provided over the proposed infrastructure are below.

3.3.1 CTP Requests/Responses

- A user asks for information relating to the CTP; that is, the user sends the CTP an initiation request. Once the initiation request is approved, the CTP's EoTs are requested by the user using HTTP requests.

- The LTS/RqMg server works as an RqMg. EoT bits are asserted according to the request's requirements.

- The request is sent to a GTS, which acts as a router. It tells the request where should it go.

- The CTP request is received at a cloud, and an RsMg is responsible for making a response.

- A response is sent from the RsMg to the RqMg through the GTS, then delivered to the user.

3.3.2 CAIQ Assessment Requests/Responses

- Users who previously utilised the CTP to obtain trust information about a specific service provider can assess the provider.

- An assessment request is sent with a '01 leftmost' flag from a user to the LTS/RqMg, which works as an LTS.
- A request is forwarded from the LTS to a GTS, which asks a CAIQ engine for the CAIQ.
- The CAIQ is sent to the user through the GTS and LTS.
- The user completes the CAIQ. A trust value is calculated and stored in the LTS, and a copy of the value is sent to the GTS. The LTS has the trust entries regarding all providers assessed by members of the user's group, and the GTS has the trust entries regarding all providers assessed by members in all groups, including updated trust values.

3.3.3 Trust Requests/Responses

- A cloud user asks for a list of all the providers that offer a specific service. This request has one of two destinations: the user's LTS or a GTS.
- The LTS directly sends a response if there is at least one entry in the LTS's table with an accepted Boolean trust (true) value and if the request does not time out.
- If the LTS has no answer, if the LTS does not have a Boolean trust value or if the request times out, the request is forwarded to the GTS, which answers it.
- Every time such a request is made, all entries in the LTS's table that relate to requests that timed out are updated using information from the GTS and an update request.

3.4 Summary

The infrastructure we proposed in this chapter is simple. It enables cloud users to use the CTP to obtain information about a specific service. Specifically, it enables users to request trust information about a specific service. Additionally, it allows providers and users to complete assessments by requesting questionnaires and completing them.

Chapter 4

Trust Assessment Techniques

In this chapter, we present different assessment techniques to calculate trust. First, we introduce a self-assessment technique that can be used by a provider. Second, we discuss cloud user assessments and their impacts on initial trust values generated by provider self-assessments. Third, we introduce the concept of weighted assessments, with which every cloud user can define his or her own assessment scheme. Fourth, we show how new users can ask for recommendations from expert users who previously completed assessments.

4.1 Introduction

As described in Chapter 2, the CTP provides a way for each user to request evidence or certificates from a cloud service provider regarding the operation of a specific service. This information gives the user a complete view of what he or she should expect while running the service. Then, after using the service, he or she may have evidence, based on experimentation, that the service was done according to the description given by the provider. Alternatively, he or she may have evidence that the service differs from the description and violates the SLA agreement between the cloud provider and the user. Based on this, we aimed to give each cloud provider the ability to assess his or her own service (i.e., to complete a self-assessment), as per (22), and to give each user the ability to assess each service after using it (i.e., to complete a feedback assessment), as per (5). Together, the two assessments make output trust values reliable by describing not only the service specifications noted in each provider's self-assessment but also the real behaviour of each service noted in each user's assessment.

4.1.1 Self-Assessments and User Assessments

As aforementioned, there are two types of assessments. The first type is self-assessment. It enables each cloud service provider to assess his or her own service behaviour. Such assessments were discussed in many papers using a normal assessment CAIQ; however, the trust values calculated using this type of assessment reflect only each service provider's view, not each client's; see (9; 18; 22). Thus, only using this type of assessment creates a one-way trust relationship between each provider and each client, which is unacceptable.

Trust relationships must also be based on customer feedback in regards to each customer's use of a service offered by a cloud service provider. This can be gathered using the second assessment type, user assessment. It is done by clients and is based on their experiences with services. It should be noted that the Cloud Security Alliance (CSA) did not make specific CAIQ questions for cloud consumers. However, Smals ICT for Society generated two cloud security assessment models that could be used by clients, whether they were normal clients or experts; see (44). The first was a normal client assessment model that could be used to compare the specifications for a service that a client needed with the specifications offered by the service's provider. The second was an expert client assessment model that enabled a cloud consumer to assess the security level of a cloud service offered by a cloud service provider. Experts could do this by answering questions in a questionnaire, such as an MCQ, covering four main characteristics expected in cloud services: governance, IAM, ITsecurity and operational security, as per (44). In this chapter, we use the expert client assessment model and select questions that can be answered with 'Yes' or 'No'. A list with all the questions and their classifications is in Appendix B.

4.2 Related Work

A cloud security requirement assessment was discussed in a previous study. The assessment gave providers the ability to complete self-assessments using cloud auditors, which are responsible for receiving service certificates from providers and for answering CAIQs; see (9). These activities could be done in two steps.

The first step was redefining 'Yes' and 'No' questions in a CAIQ using a goal question metric (GQM) approach. The approach could make the questionnaire descriptive in regards to details that could be gathered with suitable quantitative answers. Moreover, the approach used CAIQ questions generated by the CSA to obtain information on an

operational level, while the approach's goal was to measure the security levels of services. The second step was using a weighted scoring model to calculate the security levels of services offered by providers; see (15).

(60) proposed a service quality model that was based on several factors. These factors were each provider's trustworthiness, technical QoS and charges for services offered. Calculations of QoS were based on the fuzzy set theory, which is often used in regards to answers to sets of questionnaires. In the study, sets of questionnaires were defined in order to give each consumer the ability to assess each service provider based on competence, benevolence and service attribute integrity.

(22) proposed a structure for TR systems. Such systems are used to calculate the trustworthiness of a service provider based on a self-assessment done by the provider. The authors' proposed system used different attributes, such as compliance, data governance and information security, to identify each cloud provider's trustworthiness. Calculations of trust were done using and operators between and opinions were generated for in regards to different trust domains. Each provider could assess his or her own services using 'Yes' and 'No' CAIQ questions. Various questionnaires were divided into 11 domains each. The domains were represented by opinions consisting of triple values; t denoted each average rating, c denoted certainty and f denoted each initial expectation that was initialised as 0.99.

A technique was compared to the technique of (22) to calculate the trustworthiness of the authors' technique. A similarity between the techniques was that both of them used CAIQs that were initiated by the CSA to complete provider self-assessments. Additionally, both used multiple attributes as categories for MCQ questions, and an algorithm was used to determine the expectations regarding the trustworthiness of providers.

In our technique, we use consumer assessments to verify assessments done by cloud auditors. That is, we use an MCQ trust model provided by Smals ICT for Society to gather the opinions of cloud users. Moreover, our technique is based not only on expectations of providers but also on users' previous experiences with services.

As aforementioned, (22) proposed a TR system structure that represented domains using opinions consisting of triple values; specifically, $o = (t, c, f) \in \{[0, 1] \times [0, 1] \times [0, 1]\}$. Each opinion consisted of an average rating, or t; certainty, or c and an initial expectation regarding trust of a given service, or f. The authors also used a fixed value for each initial expectation, $f = 0.99$, by assuming that any service that was trusted at the beginning did not involve real assumptions; that is, there could be some untrusted or trial offers related to the service that could violate assumptions of trust.

For our study, we preferred to use a binomial subjective opinion representation, as per (31), to represent each user's opinion of each service. The opinion of a cloud user regarding a service, x, was denoted by $\omega_x = (b_x, d_x, u_x, a_x) \in \{[0,1]\times[0,1]\times[0,1]\times[0,1]\}$. It consisted of four values: b_x, a belief mass; d_x, a disbelief mass; u_x, an uncertain mass and a_x, a base rate. For opinion representations, (22) used specific kinds of logic. Similarly, our proposed technique used subjective logic to calculate trustworthiness. This was because subjective logic is the most appropriate kind of logic to use when considering uncertainty, and it is the best way to consider source trust when relatively unreliable sources are being used; see (31).

In CAIQs—for example, those discussed by (9; 18; 22)—and questionnaires provided by Smals ICT for Society—for example, one discussed by (44)—MCQ questions are divided into various domains (referred to as attributes in this paper). Each attribute considers different specifications, such as privacy, security and data governance. Therefore, each attribute is independent of the other attributes. Moreover, each attribute is divided into different sections (referred to as sub-attributes in this paper). Groups of sub-attributes can be categorised according to single domains, so they have partially dependent relationships. Each sub-attribute contains more than one MCQ question.

(22)'s method for calculating trustworthiness is different from ours. The study used 'and' operations to link attributes in order to obtain final provider opinions and to generate an expectation value, or E. For our technique, We used 'and' operations between each sub-attribute opinion to determine such opinions. Then, the opinions were aggregated by consensus operators to generate an overall user opinion. The user opinion was then visualised using barycentric coordinates, which were classified into one of six rating classes that represented the satisfaction level of a user towards a service. Next, based on a detected rating class, an ageing factor was determined that directly affected an aggregated trustworthiness value for a service from all users who completed assessments of the service, as well as for self-assessment results from the service's provider.

(41) proposed a framework for a domain-based trust model. For each domain in the framework, a trust agent was responsible for changing the domain's trust. This model was designed in order to solve security demands in a cloud.

(33) discussed how measures generated from multinomial opinions, such as trustworthiness and reliability opinions, affect reputation systems. The authors collected rates and aggregated those rates with ages. Moreover, the authors discussed how to convert continuous rating systems to discrete systems, which are suitable for Bayesian reputation systems. The fuzzy set theory was proposed for such transformations.

(66) addressed a problem regarding effects on reputation systems. That is, the authors noted that reputation scores can be altered by dishonest opinions from agents. The authors proposed a statistical filter, based on a beta distribution, to filter out unfair ratings.

4.3 Trust Assessment Approaches

For this thesis, we made assessments based on 'Yes', 'No' and 'Unknown' answers to questionnaires. As aforementioned, two types of assessments were used: provider self-assessments and cloud user assessments.

4.3.1 Provider Self-Assessments

Provider self-assessments were used to focus on users' assessments and how such assessments affect trust relationships between providers and consumers. We used a provider self-assessment technique used by (22) to generate initial trust values.

4.3.2 Cloud User Assessments

Consumer, or user, assessments were used to evaluate services offered by providers. A 'Yes' and 'No' questionnaire generated by Smals ICT for Society, mentioned in Section 4.1.1, was used to develop an overall understanding of each consumer's experience using the service. Consumer questionnaire answers were represented using binomial subjective opinions, as discussed in Section 2.4.1. Each binomial opinion was visualised with barycentric coordinates in order to categorise each opinion by rating class; this was done to determine each opinion's rating factor, which was needed for updating each initial trust value.

As aforementioned, for the questionnaire we used for cloud user assessments, shown in Appendix B, questions were classified into four different domains, or attributes: governance, IAM, IT security and operational security. Each attribute contained sub-attributes (which assessed certain properties) that contained multiple-choice questions.

Opinion Representations

We used binomial opinions to represent providers' and users' opinions. A provider's opinions about a service, x, was denoted by ω_x, while a users' opinion, A, about the service, x, was denoted by ω_x^A. A binomial opinion denoted by $\omega_x = (b_x, d_x, u_x, a_x)$.

To gather opinions with user (A) questionnaire answers, we represented sub-attributes as different opinions, where $\omega_x^{A(i,j)}$ was a binomial opinion corresponding to a sub-attribute (j) inside the attribute i regarding a service (x). $\omega_x^{A(i)}$ represented each binomial opinion collected from each attribute (i) regarding each service (x).

Gathering Opinions with Questionnaires

Opinions were gathered in regards to questionnaires using several steps. The first step was to gather sub-attribute opinion values. Each user's overall sub-opinion could be calculated using each questionnaire, as follows. First, for each sub attribute, $\omega_x^{A(i,j)} = (b_x^{A(i,j)}, d_x^{A(i,j)}, u_x^{A(i,j)}, a_x^{A(i,j)})$ was calculated based on 'Yes' and 'No' answers to a questionnaire, as well as 'Unknown' answers.

$$b_x^{A(i,j)} = \frac{p}{p+n+l}$$
$$d_x^{A(i,j)} = \frac{n}{p+n+l}$$
$$u_x^{A(i,j)} = \frac{k}{p+n+l}$$
$$a_x^{A(i,j)} = \frac{1}{2}$$

.

The elements of the above equation were as follows.

- An attribute (domain) number was represented by $i \in \mathbb{N} \wedge i \in [1, 4]$.
- A sub-attribute number was denoted by $j \in \mathbb{N}$. A first attribute contained four sub-attributes, while a second attribute contained three sub-attributes. Additionally, a third attribute contained five sub-attributes, and a fourth attribute contained three sub-attributes. Each sub-attribute comprised more than one question, for a total of $p + n + l$ questions.
- p was the number of 'Yes' answers to the sub-attribute questions.
- n was the number of 'No' answers to the sub-attribute questions.
- l was the number of 'Unknown' answers to the sub-attribute questions. (17)

The second step was to generate opinion values for the attributes. Since all sub-attributes had common assessment criteria (e.g., security), we could not consider the

sub-attribute opinions, $\omega_x^{A(i,j)}$, as independent of each other. Therefore, an average fusion operator was used to calculate the attribute opinions: $\omega_x^{A(i)} = \underline{\oplus}_{j \in R(i)} \omega_x^{A(i,j)}$.

To elucidate, this type of assessment generated four binomial opinions that corresponded to four different domains, or attributes. The generated attribute opinions were $\omega_x^{A(1)}$, $\omega_x^{A(2)}$, $\omega_x^{A(3)}$ and $\omega_x^{A(4)}$, such that:

$$\omega_x^{A(1)} = \omega_x^{A(1,1)} \underline{\oplus} \omega_x^{A(1,2)} \underline{\oplus} \omega_x^{A(1,3)} \underline{\oplus} \omega_x^{A(1,4)}$$

$$\omega_x^{A(2)} = \omega_x^{A(2,1)} \underline{\oplus} \omega_x^{A(2,2)} \underline{\oplus} \omega_x^{A(2,3)}$$

$$\omega_x^{A(3)} = \omega_x^{A(3,1)} \underline{\oplus} \omega_x^{A(3,2)} \underline{\oplus} \omega_x^{A(3,3)} \underline{\oplus} \omega_x^{A(3,4)} \underline{\oplus} \omega_x^{A(3,5)}$$

$$\omega_x^{A(4)} = \omega_x^{A(4,1)} \underline{\oplus} \omega_x^{A(4,2)} \underline{\oplus} \omega_x^{A(4,3)}$$

.

The third step was to generate overall user opinion values. Each attribute assessed a service from a different point of view, so the attribute opinions could be considered independent of each other. Therefore, the overall opinions could be generated using cumulative fusion operators (also known as independent consensus operators) between the four attribute opinions:

$$\omega_x^A = \oplus_{i \in [1,4]} \omega_x^{A(i)} = \omega_x^{A(1)} \oplus \omega_x^{A(2)} \oplus \omega_x^{A(3)} \oplus \omega_x^{A(4)}$$

The fourth step was to classify opinions. This step was used to classify the overall user opinion generated from the questionnaire answers used for the user assessment. The aim was to classify a binomial user opinion, ω_x^A, into a finite number of fuzzy-meaning classes. We assumed that we had seven classes: very good, good, very bad, bad, low good, low bad and very uncertain. Additionally, we needed to visualise each user opinion so that each tester could determine whether a classifier worked correctly or not, as shown in Chapter 6.

Binomial opinions were visualised using the barycentric coordinates introduced in Section 2.4.1 to classify each opinion. We split areas inside triangles into sub-areas; each sub-area had properties that were similar to all points in it. For the barycentric coordinates, if lines could be drawn from the middle points between two triangle sides and were parallel to opposite triangle sides, we constructed an inner reversed triangle with three sides, described using $u_x = 0.5$, $b_x = 0.5$ and $d_x = 0.5$ and shown in Figure 4.1.

Therefore, if an opinion was in an orange triangle, belief regarding this opinion was

Figure 4.1: The binomial opinion rating classification.

greater than 0.5, while disbelief and uncertainty were less than 0.25. This meant that the opinion was positive. However, if the opinion was in a blue triangle, it was negative ($d_x > 0.5$), and if the opinion was in a green triangle, it was uncertain ($u_x > 0.5$). Similarly, if an inner triangle could be divided as in Figure 4.1, and if the opinion was in a grey triangle, it was negative (although less so than if it was in a blue triangle), and if the opinion was in a yellow triangle, it was positive (although less so than if it was in an orange triangle).

Then, we assumed that we had six rating classes for the opinion in the triangle, as shown in Figure 4.1: very good, good, very bad, bad, unnamed (later split into low good and low bad) and very uncertain classes. These classifications were based on the values belief, b_x; disbelief, d_x and uncertainty, u_x. Table 4.1 shows the ranges of these three variables for inside each region. We defined a variable, k, that represented the scalar value of each rating class. The class rating k was a rational value that ranged from $+1$, or very good, to -1, or very bad. Moreover, good and low good were $k = +1/2$ and $k = +1/4$, respectively, while bad and low bad were $k = -1/2$ and $k = -1/4$, respectively. Finally, uncertain was $k = 0$.

The fifth step was to update trust values. The aim of this step is to show how a user opinion classification (from step four) updated a trustworthy scalar value stored for a service (x). Below, we present two approaches to updating trust. They are using only aggregation and using aggregation with ageing operations.

We assumed that each provider had the ability to complete a first assessment regard-

Region	Belief	Disbelief	Uncertainty
Very good, certain	$b_x \geq 0.5$	$d_x < 0.5$	$u_x < 0.5$
Good, certain	$0.25 < b_x < 0.5$	$d_x < 0.25$	$u_x < 0.5$
Very bad, certain	$b_x < 0.5$	$d_x \geq 0.5$	$u_x < 0.5$
Bad, certain	$b_x < 0.25$	$0.25 < d_x < 0.5$	$u_x < 0.5$
Unnamed, certain	$0.25 \leq b_x < 0.5$	$0.25 \leq d_x < 0.5$	$u_x < 0.5$
Very uncertain	$b_x \leq 0.25$	$d_x \leq 0.25$	$u_x \geq 0.5$

Table 4.1: The classification rules.

ing his or her service by answering a CAIQ, which produced an initial scalar trust value. We could use a technique used by (22) to generate an initial trust value, or we could use steps one to four. This could be done by collecting a provider's opinion using the approach in Section (steps one to three). Then, the opinion could be classified into a rating class using the classifier described in step four, and a class rating value, k, would be obtained. Finally, k, which could range from -1 to +1, would be scaled to another scalar value ranging from 0–100, respectively.

Each cloud user had the ability to reassess a service after using it and submit his or her opinion about the service's operation in a questionnaire, as mentioned in Section 4.3.2. The user's opinion would update the service's scalar trust value, either through only aggregation or through aggregation with ageing. The update would depend on which rating class the user's opinion was classified as.

Using only aggregation. There are many aggregation methods. For this study, aggregation was done with simple addition. Specifically, an aggregation constant, $\lambda \in [0, 1]$, was used. Aggregation had no effect on an original rating if $\lambda = 0$, while it had a significant effect if $\lambda = 1$.

The following examples use only one user, A; later, the formula for any number of users is shown. First, assume that any user can do any positive number of assessments

of one service at different times. Second, consider the following definitions.

- $R_{x,(t=0)}$ is the initial rating value (from only a provider) generated from the provider's self-assessment for service x at the beginning ($t = 0$).
- $R^A_{x,(t\neq 0)}$ is an old rating value (from the provider and user A) from over time t for service x.
- $R^A_{x,(t+1)}$ is the overall new accumulated rating value (from the provider and user A) after time period $t + 1$ for service x.
- $R_{x,(t+1)}$ is the overall new accumulated rating value (from provider and all users) after time period $t + 1$ for service x.

In order to give permission to any user to do assessments any number of times, our method for calculating a reputation (rating) value, generated from any agent A towards service x, depended not only on the current opinion rating class k_{t+1} but also on the previous opinion rating class k_t. The rationale for completing another assessment was to remeasure the reputation and produce a new value instead of an old one. Thus, our method was based on updating an overall reputation value with a new opinion and removing the old opinions of all users who did multiple assessments.

Assume that the value of a previous opinion rating class for agents who completed initial assessments is $k_t = 0$ (neutral value). The new accumulated rating, $R^A_{x,(t+1)}$, after time period $t + 1$ can be expressed as follows. For a first user assessment,

$$R^A_{x,(t+1)} = \lambda' + R_{x,(t=0)}$$

,

where

$$\lambda' = (k_{t+1} - k_t)\lambda,\ 0 \leq \lambda \leq 1$$

.

For any user assessment except the first assessment,

$$R^A_{x,(t+1)} = \lambda' + R^A_{x,(t\neq 0)}$$

,

where

$$\lambda' = (k_{t+1} - k_t)\lambda$$

.

Let symbol $\mathbb{A}$ represent a set of all the users who assessed service x. The overall reputation (rating) is generated from all the users average overall ratings, as follows.

$$R_{x,(t+1)} = \frac{\sum\limits_{A \in \mathbb{A}} R^{A}_{x,(t+1)}}{|\mathbb{A}|}$$

.

Then, assume that the overall reputation $R_{x,(t+1)}$ has a low bound of 0 and a high bound of 100. If the calculated overall reputation is out of the boundaries, modify it so it is in the boundaries (0, if it is modified to be smaller, and 100, if it is modified to be bigger).

Using aggregation with ageing. The first method for collecting users' opinions (using only aggregation) depends only on the last assessment of each user and removes all history. The second method for collecting users' opinions is aggregating the last assessment outcome of a user with an age value for the history generated by the user. For this method, ageing factor is defined as $\Lambda \in [0, 1]$. The value of Λ determines a history ratio for the user's opinions that, with a new opinion, generates the current reputation value of the user in regards to any service. However, the history is removed, as in the previous method, if $\Lambda = 0$, and the history contributes to the ratio if $\Lambda = 1$.

For service x, assume that user A can do any number of assessments. Therefore, if user A has only one assessment, there is no need for ageing because that user has no assessment history. However, if the user has more than one assessment and they were completed at different times, the most recent assessment's effect will be aggregated in regards to the age effect of the user's assessment history. Define $\delta^{A}_{x,t+1}$ as the most recent update effect at time $t + 1$ and define $\delta^{A}_{x,t}$ as the update effect of all assessments in time $[0, t]$ for every user A that assessed service x. The update value can be calculated as follows.

For the first user assessment, there is no assessment history for user A regarding service x. Therefore, there is no need for ageing. λ is an aggregation constant:

$$\delta^{A}_{x,t+1} = k_t \lambda$$

.

However, for any assessment except the first assessment, there is an assessment history for user A. Therefore, we should use an ageing factor:

$$\delta^{A}_{x,t+1} = k_t\lambda + \Lambda\delta^{A}_{x,t}$$

To decrease the effect of the history, use $\Lambda = 0.01$ (very close to 0). To increase the effect of history on the calculation of a reputation value, use $\Lambda = 0.99$ (very close to 1).

Then, note that for an overall reputation (rating) generated from all users $A \in \mathbb{A}$ for service x, $\mathbb{A}$ is a set of all users whose assessments are a sum of all updates done by the users plus initial trust $R_{x,(t=0)}$, generated from a provider self-assessment. This is shown as follows:

$$R_{x,(t+1)} = \sum_{A\in\mathbb{A}} \delta^{A}_{x,t+1} + R_{x,(t=0)}$$

.

Subsequently, assume that the overall reputation, $R_{x,(t+1)}$, has a low bound of 0 and a high bound of 100. If the calculated overall reputation is outside the boundaries, modify it so it is inside the boundaries (0 if it needs to be smaller and 100 if it needs to be bigger).

For example, suppose user A utilises service x. The service has the initial rating $R_{x,t}$ at time t. Based on the user's experience while using this service, he or she does three assessments.

The first is done at time $t + 1$ with an outcome opinion category of low good, which means that $k_{t+1} = 0.25$ is the initial value for $k_t = 0$. Using only aggregation, the overall aggregated rating value for service x is

$$R^{A}_{x,(t+1)} = 0.25 \times \lambda + R_{x,t}$$

.

In regards to using aggregation with ageing, note that this is the first assessment for user A regarding service x, so there is no assessment history and there is no need to use ageing:

$$\delta^{A}_{x,t+1} = 0.25 \times \lambda$$

$$R^{A}_{x,(t+1)} = 0.25 \times \lambda + R_{x,t}$$

.

The second assessment is done when the user finds that the service is not good, unlike before. He or she completes another assessment at time $t + 2$ with the overall

agent opinion class very bad; $k_{t+2} = -1$. Using only aggregation,

$$R^A_{x,(t+2)} = (-1 - 0.25) \times \lambda + R^A_{x,(t+1)}$$
$$R^A_{x,(t+2)} = -1.25\lambda + 0.25\lambda + R_{x,t} = -\lambda + R_{x,t}$$

.

Using aggregation with ageing,

$$\delta^A_{x,t+2} = -\lambda + 0.25 \times \lambda \times \Lambda$$
$$R^A_{x,(t+2)} = (-1 + 0.25\Lambda)\lambda + R_{x,t}$$

.

The third assessment is done when the provider sees that his or her overall user rating has decreased. The provider enhances the service and asks the user to do another assessment at time $(t+3)$.The user's opinion is that the service is good, and $(k_{t+3} = 0.5)$. Using only aggregation,

$$R^A_{x,(t+3)} = (0.5 - (-1)) \times \lambda + R^A_{x,(t+2)}$$
$$R^A_{x,(t+3)} = 1.5\lambda - \lambda + R_{x,t} = -\lambda + R_{x,t}$$

.

Using aggregation with ageing,

$$\delta^A_{x,t+3} = 0.5\lambda - \lambda \times \Lambda + 0.25 \times \lambda \times \Lambda^2$$
$$R^A_{x,(t+3)} = (0.5 - \Lambda + 0.25\Lambda^2)\lambda + R_{x,t}$$

.

4.3.3 Unequal Weighted Assessments

The previous approaches addressed opinions collected from the attributes in equally weighted ways in order to generate overall user opinions. This means that if there were four attributes (governance, IAM, IT security and operational security), each one would make a 25% contribution to each overall user opinion. However, there are some services with which security considerations, for example, are the most important factors that need to be assessed. In such situations, a method needs to be used that allows the

utilisation of different weights based on service characteristics.

In such a method, after a cloud user answers an MCQ, she or he is asked about weight ratios for each attribute before an overall opinion is calculated. Specifically, the user is asked for four ratios that correspond to four domains (attributes). These weights are applied to an opinion-calculation methodology with a version of an independent cumulative fusion operator that supports different weights. The operator is discussed below.

A Weighted Consensus between Independent Opinions

Three cases are discussed in this section. To begin, let there be two independent opinions for agents A and B regarding service x. The agents A and B have the opinions $\omega_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$ and $\omega_x^B = (b_x^B, d_x^B, u_x^B, a_x^B)$, respectively, about common service x. Assume that the weights of these opinions are α and β for ω_x^A and ω_x^B, respectively.

Then, assume that $\eta\omega_x^{A\triangledown B}$ represents the overall weighted opinion of both agents A and B regarding service x with the overall weight η. The weighted consensus operation symbol is $\boxplus$. Subsequently, let $\omega_x^{A\triangledown B} = (b_x^{A\triangledown B}, d_x^{A\triangledown B}, u_x^{A\triangledown B}, a_x^{A\triangledown B})$. It can be calculated as per (69).

For the first case, if $(u_x^A \neq 0 \wedge u_x^B \neq 0) \vee (u_x^A \neq 1 \wedge u_x^B \neq 1)$, then

$$b_x^{A\triangledown B} = \frac{(k - u_x^A u_x^B)(\alpha b_x^A u_x^B + \beta b_x^B u_x^A)}{k(\alpha u_x^B + \beta u_x^A - (\alpha + \beta) u_x^A u_x^B)}$$

and

$$d_x^{A\triangledown B} = \frac{(k - u_x^A u_x^B)(\alpha d_x^A u_x^B + \beta d_x^B u_x^A)}{k(\alpha u_x^B + \beta u_x^A - (\alpha + \beta) u_x^A u_x^B)}$$

.

Then,

$$u_x^{A\triangledown B} = \frac{u_x^A u_x^B}{k}$$

and

$$a_x^{A\triangledown B} = \frac{\alpha a_x^A u_x^B + \beta a_x^B u_x^A - (\alpha a_x^A + \beta a_x^B) u_x^A u_x^B}{\alpha u_x^B + \beta u_x^A - (\alpha + \beta) u_x^A u_x^B}$$

,

where $k = u_x^B + u_x^A - u_x^A u_x^B$; see (69). Note that in our system, we use $a_x^A = u_x^B = 1/2$. Therefore, the resulting base rate is $a_x^{A\triangledown B} = 1/2$, no matter the values of weights α and

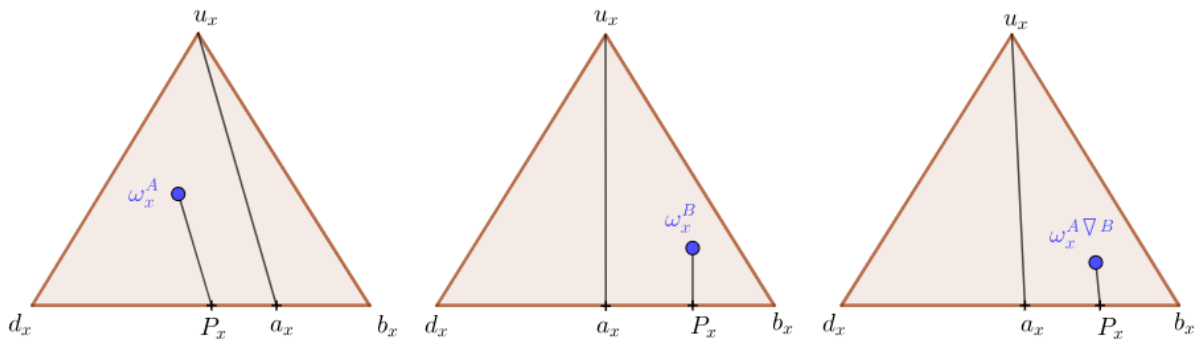

Figure 4.2: An example of a weighted consensus operation for two independent binomial opinions with the weights $\alpha = 1$ and $\beta = 2$, respectively, applied to ω_x^A and ω_x^B, respectively.

β.

An example of a consensus between two independent binomial opinions is shown in Figure 4.2. The first opinion is $\omega_x^A = \{0.25, 0.35, 0.4, 0.75\}$, and it has a projected probability of 0.55. The second opinion is $\omega_x^B = \{0.67, 0.1, 0.23, 0.5\}$, and it has a projected probability of 0.785. An opinion is obtained after a consensus with a weight of $\alpha = 1$ is applied to the opinion ω_x^A and after a consensus with a weight of $\beta = 2$ is applied to the opinion ω_x^B. The obtained opinion is $\omega_x^{A\nabla B} = \{0.652, 0.178, 0.17, 0.55\}$, and it has a projected probability of 0.715, which means that opinion ω_x^B influenced the result of opinion ω_x^A twice.

For the second case, if $(u_x^A = 0 \wedge u_x^B = 0)$, the two opinions are $\omega_x^A = (b_x^A, d_x^A, 0, 1/2)$ and $\omega_x^B = (b_x^B, d_x^B, 0, 1/2)$, where $b_x^A + d_x^B = 1$ and $b_x^B + d_x^B = 1$.

$$b_x^{A\nabla B} = \frac{\alpha b_x^A + \beta b_x^B}{\alpha + \beta}$$

The equation above is the average of α times b_x^A and β and times b_x^B. The result satisfies the equation $\min(b_x^A, b_x^B) \leq b_x^{A\nabla B} \leq \max(b_x^A, b_x^B)$. As $0 \leq b_x^A \leq 1$, $0 \leq b_x^B \leq 1$. We can conclude that $0 \leq b_x^{A\nabla B} \leq 1$:

$$d_x^{A\nabla B} = \frac{\alpha d_x^A + \beta d_x^B}{\alpha + \beta}$$

.

Then, similar to the explanation above, we can conclude that $0 \leq d_x^{A\nabla B} \leq 1$. However, we need to prove that $b_x^{A\nabla B} + d_x^{A\nabla B} = 1$:

$$b_x^{A\triangledown B} + d_x^{A\triangledown B} = \frac{\alpha b_x^A + \beta b_x^B + \alpha d_x^A + \beta d_x^B}{\alpha + \beta} = \frac{\alpha(b_x^A + d_x^A) + \beta(b_x^B + d_x^B)}{\alpha + \beta}$$

.

As $b_x^A + d_x^A = 1$ and $b_x^B + d_x^B = 1$, $b_x^{A\triangledown B} + d_x^{A\triangledown B} = 1$:

$$u_x^{A\triangledown B} = 0$$

$$a_x^{A\triangledown B} = 1/2$$

.

Finally, for the third case, if $(u_x^A = 1 \wedge u_x^B = 1)$, the two opinions are $\omega_x^A = (0, 0, 1, 1/2)$ and $\omega_x^B = (0, 0, 1, 1/2)$:

$$\omega_x^{A\triangledown B} = (0, 0, 1, 1/2)$$

.

Generating Users' Opinions with Weights

Assume that the weights for the four attributes are denoted by $\alpha_1 > 0, \alpha_2 > 0, \alpha_3 > 0$ and $\alpha_4 > 0$. Then, follow three steps.

For steps one to two, use the same method as before to extract sub-attribute opinion values and determine the attributes' opinions using averaging operators to generate the four domains' opinions, which are $\omega_x^{A(1)}, \omega_x^{A(2)}, \omega_x^{A(3)} and \omega_x^{A(4)}$.

For step three, generate an overall weighted opinion, as follows:
$\omega_x^A = (\alpha_1\omega_x^{A(1)} \boxplus \alpha_2\omega_x^{A(2)}) \oplus (\alpha_3\omega_x^{A(3)} \boxplus \alpha_4\omega_x^{A(4)})$.

The overall user opinion represents a normal consensus operator between two independent opinions. The first opinion is generated using a weighted consensus between the first two attributes' opinions with the weights α_1 and α_2. The second opinion is generated using a weighted consensus between the last two attributes' opinions with the weights α_3 and α_4.

4.3.4 Assessments with Transferred Trust Decisions

There are users who are not experts and cannot answer MCQs, and those users have a right to assess services. Instead of forcing the users to complete blind assessments, which is unfair, give those users the ability to complete assessments with help from users they

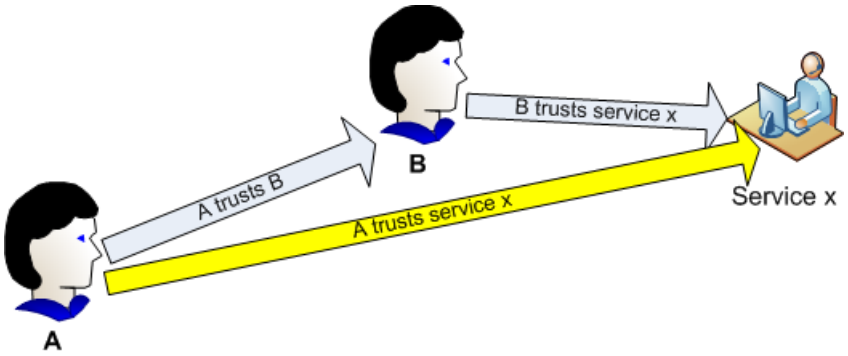

Figure 4.3: Opinion transitivity.

trust who have already completed assessments. Doing this is similar to transferring trust between users and propagating recommendations from old users to new users completing assessments. Give the users two options: complete their own assessments, similar to how users complete assessments before, or ask other cloud users who have already done assessments to do the new users' assessments instead.

In this section, we suggest a method for generating users' overall opinions using a fusion-trust transitivity operator. Specifically, we introduce a trust transitivity operator. Later, we present our suggestion for propagating opinions from old users to new users.

Trust Transitivity

As in Figure 4.3, suppose that there are two different agents, A and B, and one service, x. Agent B has a clear opinion about service x that is denoted by ω_x^B, but agent A does not have a clear opinion. However, agent A trusts agent B, and this trust is represented by his or her opinion of agent B, which is represented by ω_B^A. Using transitivity, agent A's opinion of service x can be predicted by transferring B's opinion to agent A using the trusting relationship between agents A and B. The transitivity operation can be represented by the symbol $\otimes$. The transferred opinion cane denoted by $\omega_x^{A:B} = \omega_B^A \otimes \omega_x^B = (b_x^{A:B}, d_x^{A:B}, u_x^{A:B}, a_x^{A:B})$.

The transitivity can be done using three methods. The first method is uncertainty favouring discounting, which can increase the uncertainty of the transferred opinion. The second method is base rate sensitive transitivity, which can be primarily affected by the expectation value of agent A regarding agent B. Finally, the third method is opposite belief favouring, which is based on the adage 'the enemy of my enemy is my friend'. This method is the most suitable for our system; see (34).

Opposite belief favouring. An overall opinion generated using the opposite belief

favouring method, $\omega_x^{A:B} = (b_x^{A:B}, d_x^{A:B}, u_x^{A:B}, a_x^{A:B})$, can be calculated using a formula posited by (34):

$$b_x^{A:B} = b_B^A b_x^B + d_B^A d_x^B$$

$$d_x^{A:B} = b_B^A d_x^B + d_B^A b_x^B$$

$$u_x^{A:B} = u_B^A + (b_B^A + d_B^A) u_x^B$$

$$a_x^{A:B} = a_x^B$$

Note that $b_B^A, d_B^A and u_B^A$ are belief, disbelief and uncertainty values for A's opinion of B.

Generating User Opinions Based on Trust Transference

To transfer opinions between cloud agents, or users, store the following information for every cloud node A_i. First, store the cloud user's opinion, A_i, regarding service x, which is denoted by $\omega_x^{A_i}$. Second, store a set of all cloud users that helped cloud user i generate his or her opinion regarding service x. L denotes the set, and $|L|$ denotes the number of elements inside the set. If $|L| = 0$, then agent $i$generated his or her own opinion $\omega_x^{A_i}$ without any help.

With our approach, a new user can send requests to old users (i.e., users who completed assessments previously) to send him or her their opinions about a service that are already stored. We refer to old users as 'helping users'. The new user must submit his or her opinion of the service to the helping users. The new user's request is represented by a directed tree, in which a top parent (the top of the tree) represents the new user.This can be shown using two cases.

For the first case, suppose that each new user can request to transfer trust from only one old user (i.e., assume that $|L| = 1$). The opinion of agent A, the old user, regarding service x, which helps agent B, the new user, is denoted by $\omega_x^{A:B} = \omega_B^A \otimes \omega_x^B$. For the second case, suppose that any new user, A, can transfer trust from multiple helping users (i.e., assume that $|L| > 1$). The opinion of user A regarding service x, which helps users inside the set L, can be calculated using two steps. First, for every user $B_i \in L$, where $i \leq |L|$, calculate the transferred opinion $\omega_x^{A:B_i} = \omega_{B_i}^A \otimes \omega_x^{B_i}$ using the opposite belief favouring method.Second, in order to generate the overall opinion of user A with help from the users inside the set L, conduct an averaging operation for all $|L|$ calculated opinions using step one. Leaves represent the helping users who completed assessments

without help. Other parentsrepresent helping users who transferred generated opinions using transitivity.

4.4 Summary

The concept of a self-assessment completed by a provider using a CAIQ is not new, but each trust value generated with such an assessment reflects only cloud providers' opinions of their own services. This is unfair as trust requires two elements: a provider and a customer. Therefore, in this section, we introduced a method to allow a cloud users to assess any service he or she has used. We used a service assessment questionnaire provided by Smals ICT for Society. Our approach was based on collecting a user's opinion using the questionnaire and a subjective logic operation in which the opinion was denoted by a subjective binomial opinion. We also introduced a method for determining the scalar update effect of a user's opinion using a classifier that utilises barycentric coordinates. We presented two methods for determining how the user's opinion affected the most recent trust value of a service using only aggregation or aggregation with ageing. Additionally, we presented a method for determining how a user defined his or her assessment criteria by defining desired weights for main assessment domains, and we presented a method for solving this situation using weighted subjective operators. Finally, we presented a method for determining how a user transferred his or her assessment to other trusted users or experts and for determining how the user's opinion was generated using transitivity.

Chapter 5

Securing the System Against Untrusted Users

5.1 Introduction

Nowadays, blockchain technologies (8; 53; 67) are becoming increasingly important. Blockchain technology can solve different problems and make the solutions easier to apply over the internet rather than using paper work. A blockchain is a secured, trustless platform that enables complex computations. It uses public and private cryptography keys (3) to encrypt and decrypt data sent or stored over a blockchain. Operations are performed using transactions; these transactions are stored in the form of blocks, in which every node connected to a blockchain can verify whether or not a generated block is correct. Some of these nodes are called miners, which compete with each other to initiate the mining first so that they can get rewards back in the form of Ether (a cryptocurrency).This digital asset allows miners to use their computational resources from the mining to generate new blocks and extend the blockchain.

The unique characteristics of blockchains encouraged many, especially computer scientists, to consider offering services over this technology to benefit from this secure and trustless platform(53; 61). In this chapter, we will list some of the services that can be improved using blockchain technology and present our idea of how we can use blockchains to remove unfair cloud user assessments. We present two ways to accomplish this. First, any node in a blockchain could assess a new user. Second, only a set of trusted users could conduct assessments, and if a new user gained their trust, he or she would be added to the trusted set. Then, that individual would be allowed to assess new users later. We also developed a reward/penalty system to give users rewards as a result of

their honest assessments and apply a penalty to users who behave maliciously during the assessment process.

5.2 Why is Blockchain Technology so Important?

Blockchain technology could be an ideal way to do secure transactions over the internet without worrying about security breaches. Many cloud services can be developed using blockchain technology, such as decentralised decision applications (67; 70), peer-to-peer financial transactions (70), distributed cloud storage (40), digital identity identification (58) and smart contracts (14). All of these applications are still under development and require more research. In our work, we use blockchains to perform assessments over the cloud, removing unfair assessments to prevent them from affecting the TR system we have proposed.

5.2.1 Blockchains in Decentralised Decision Applications

For decentralised decision applications, one of the most important functions is designing an online voting system that works over the internet and enables voters to submit their votes remotely. When it comes to replacing current voting systems with an internet-based system, however, people are likely to be afraid of security breaches.

The first internet voting process was done in Estonia in 2013. (59) presented a security analysis of the new Estonian voting system, identifying a problem that occurred when the government was preparing the election software: The official voting software keys were downloaded over unsecured internet connections. Furthermore, theyrecorded login details within the field of view of CCTV cameras.This resulted in a leak of more than 30% of its ballots over the internet.

This problem can be resolved by designing a voting system based on blockchain technology. A voter could then submit his or her vote without worrying about whether the vote could be discovered by anyone else. This means that voters' identities would stay anonymous to other members of the blockchain. Another concern with online voting, which belongs to the domain of democracy itself, is that citizens do not want one organisation to control the election process.This can also be resolved by blockchain technology, which provides decentralised decisions.

Another decentralised decision application is for a shareholders' association(64). Let us assume that there exists a company owned by certain individuals, and each one

holds a percentage of the capital. The problem is that there are different venues in which these individuals could invest. Blockchain technology would enable us to design a system that allows members to submit investment proposals and run a voting process for these proposals, in which each member's vote is proportional to his or her percentage of shares owned.The income would also be distributed in proportion to this percentage.

5.2.2 Blockchains in Financial Applications

Using blockchain technology, we can create digital coins (6), such as Bitcoins, Ether, and other forms of cryptocurrency. A blockchain enables peer-to-peer money transfers without the aid of a third party. This resolves the problem of searching for and utilising a trusted agency or bank to do money transfers. All the transactions are secured through a public key for cryptocurrency.All nodes over the network can verify the transaction using the public key, but only the actual receiver can encode it and receive the money.

5.2.3 Blockchains in Distributed Cloud Storage

The use of traditional cloud storage gives some providers complete control over the cloud data. This may raise a concern among cloud users that their data could be stolen from the provider's servers. A benefit of using blockchain technology is that it can distribute user files between different nodes in the blockchain instead of storing all the files on only one cloud platform.This enables unlimited storage for users, but at the same time, privacy and security issues should be managed.

5.2.4 Blockchains in Digital Identity Identification

Presently, as a result of new technologies with huge computation capabilities, the incidence of data breach situations has increased, and most individuals can be affected by these breaches.

In 2013, a huge retailer called *Target* was exposed to a breach of about 70 million customers' data, including their names, addresses, emails and even their payment card numbers. The company paid $18.5 million to settle the resulting claims (54).

In (62), the most well-known data breach cases before 2015 were reviewed. According to this study, the confirmed data breach cases from 2012 to 2014 resulted in approximately 443 million records stolen from 183 retailers.

One of the most recent attacks happened on the 1st of October, 2017 when hackers

stole information about Pizza Hut customers through its website. This information included the clients' delivery addresses, email addresses, and not only their payment card numbers but also their CVV numbers and expiration dates. This data breach enabled the hackers to steal money from the affected clients (2).

Blockchain technology can prevent such data breaches by using the public key cryptography method, in which sensitive information, such as private passwords, do not need to be revealed, and other nodes can use public keys to verify the transactions.

5.3 Smart Contracts

Before explaining what a smart contract is, let us introduce a common situation in our daily lives to see how we can perform it in a better way using smart contracts and blockchains. Suppose that youwant to buy a property or an item, such as a house or a car. The traditional way to begin is to go to a lawyer to verify its documentation, and, naturally, the lawyer's advice costs you money. After that, you will likely search for an intermediate individual or company through which you can buy the property or item. The intermediary will charge a fee as well. After taking these steps, you may still feel that something could go wrong, as you may not trust the lawyer or the intermediary; the property or item might even be sold to another property owner after you have just purchased it.Now, let us consider the same case using blockchain technology. Suppose that the property or item for sale is uploaded over a blockchain. All the property or item documents are uploaded and secured on the blockchain. If you are interested in purchasing it, you may pay a deposit through a cryptocurrency transaction in order to view the documents, and you are sure that these documents are correct as they were revised by specialists when they were uploaded. In this case, you do not have to search for and pay a trusted lawyer to review the documents for you. If you decide that you will buy this property, you just need to transfer the money to the property owner, without any need for an intermediary (or the related fees for the intermediary). Once the owner receives the payment, the property ownership will be changed over to you directly; all the other nodes on the network will now be able to see that the purchase is complete and that you are the new owner. By following these steps, you will not have any worries that the previous owner will sell the property again to someone else as he or she is no longer recognised as the owner of this property on the blockchain. So, we can conclude that using blockchains can make the purchasing process faster, less expensive and more secure than before.This can be done via smart contracts, which govern all the actions

between a seller and a buyer.

Smart contracts can be referred to by different names, such as self-executing, blockchain or digital contracts. The term 'smart contract' was first introduced by Nick Szabo in 1993. It described a digital vending machine operation. Since then, the term has been used in various applications, and it is now used for blockchains. A smart contract is a piece of cryptography code that is run over a blockchain. It is a self-executed program that runs according to the terms agreed upon between the user and the creator and is specified by the contract creator. It enables money transfer transactions (in the form of Ether, as described in Chapter 2) between the user and the contractand then to and from the creator. Once a smart contract is deployed and published over the blockchain, all the blockchain users can see, validate and use the contract. No one can modify a smart contract after its deployment, which circumvents any dishonest means of changing the contract terms at a later date.

There are different platforms based on blockchains that enable the running or writing of smart contracts. Some of the most popular platforms are Bitcoin, SideChains, Nxt and Ethereum.

- Bitcoin: This platform supports bitcoin transactions, but it does not have a strong capability to process documents (27; 36; 48).

- SideChains: This platform supports bitcoin transactions as well as contract processing (52).

- Nxt: This public, open-source blockchain platform enables bitcoin transactions and supports a limited number of smart contracts, such as those for the voting system, but it still does not allow users to write their own smart contracts (1).

- Ethereum: This is a public blockchain platform. It has its own digital currency, called Ether. Ethereum has two versions: an open source one, EthereumJ, which supports Java, and an executable one, an Ethereum wallet.Bitcoin transactions are supported by Ethereum, and users can write their own smart contracts and deploy them over the blockchain. It also supports working over two networks: the main blockchain and a test network. The main network is the actual blockchain network,and any transaction should be done using real Ether. The test network is a copy of the actual network, but it uses virtual Ether for testing purposes. Smart contracts can be written over an Ethereum wallet using Solidity, a scripting language (13; 19).

5.3.1 How Is a Smart Contract Executed?

Smart contracts can be written in different programming languages. The most frequently used one is Solidity, a scripting language based on JavaScript, which enables users to deal with a blockchain and its contracts. A smart contract is a block of code that offers functions to the blockchain nodes. The smart contract structure contains not only (state) variable definitions but also functions, which perform the aim of the smart contract. Once a smart contract is written and ready for deployment, some conditions must be met so that the smart contract can be published.

A smart contract must be deployed from a valid account. In order to publish a smart contract, a fee must be paid, which is referred to by the term 'gas'. The amount of gas required to run a smart contract depends on the amount of computation required for all the functions offered by the contract. Consequently, when more functions (i.e., coding lines) are added to the smart contract, more gas needs to be paid. So, before deploying a contract, the Ethereum wallet calculates how much gas is required to execute the contract and get it published, which is described by the term 'maximum gas'.Solidity offers ways to reduce the maximum gas level as much as possible, which will be discussed in Sections 5.5 and 5.6. On the one hand, if the money (gas or Ether) in the creator's account is equivalent to (or in excess of) the maximum gas, the contract will be executed and published correctly. On the other hand, if the creator does not have enough money to cover the maximum gas required, Ethereum begins to execute the contract line by line; for every line, it consumes some gas until the creator's wallet reaches an out-of-gas state. If the creator's wallet is out of gas and the contract is not yet executed in full, the contract will not be deployed, and the gas removed from the owner's wallet will not be refunded.

A smart contract contains a wallet that stores Ether. Ether can be transferred to or from a creator, other accounts and contracts. Once a smart contract is published, all the blockchain users will see, validate and take a copy of the contract. Smart contracts, as well as transactions, are stored in the form of blocks. A smart contract cannot be modified after it is deployed. If anyone tries to modify anything that is published in a blockchain, the other nodes will detect it, which prevents that individual from changing any blocks.

5.4 Removing Unfair Assessments

In this section, we introduce the problem of unfair assessments and how they can be solved using blockchain technology. As we introduced in Chapter 4, cloud users can do assessments of the services they have used based on their experience with those services. We offered cloud users an MCQ, modelled on one created by the Smals research group in France(44). The users answered the questionnaire, and based on their answers, we were able to extract their opinions using the method suggested in Chapter 4. We also suggested a way to classify the extracted opinions into a fuzzy rating class, which described how an opinion would change the most recent trust value of the service being assessed. The aim of this chapter is to provide a pre-stage before applying the update action. The action of the pre-stage is to filter out unfair opinions before updating the TR system. The reason we need this filtering stage is that unfair assessments may have been done by users who answered the MCQ randomly; alternately, the assessment system may have been hacked, resulting in malicious assessments that could destroy the TR system and reflect incorrect trust values for the services.

We also suggest a reviewing process for MCQ answers that were submitted as an assessment for a service that the assessor has not actually used. This reviewing process will be done by the blockchain users, who can access and review the MCQ answers based on their previous experience with the service being assessed. Based on their observations, each answer is assigneda value between 0 and 1, which represents the extent to which the user was honest and fair in his or her service assessment. A value of 1 denotes that the user did the a service assessment honestly, while a value of 0 means that the user was not honest in answering the service assessment questionnaire, and his or her opinion should not update the trust value of the service in the TR system.

We assume that the blockchain users who review a certain service assessment are aware of the service function and have done assessments of that service in the past. So, these blockchain users, based on their experience with the service, know its strengths and the weaknesses and can recognise whether the answers provided by the review requestreflect those points or not. If the answers do not reflect the actual operation of the service,the blockchain users will consider the submitted service assessment to be unfair, so these answers should not be taken into consideration while updating the TR system.

Each blockchain user can submit his or her review value within a specific time frame. When the time frame expires, a decentralised counting process begins to check whether

the user assessment should be trusted or not. The assessment will be trusted and included in the update step described in Chapter 4 if the number of blockchain users giving a review of more than or equal to 0.5 is greater than or equal to half of the total number of blockchain users who did the assessment; otherwise, it will be rejected before updating the TR system. To encourage the blockchain users to do a review, a reward in form of Ether will be sent to them. However, there could be situations in which the reward process could be considered a point of weakness in our system. Problems arise when a block chain user habitually does reviews but does not pay attention to whether the review is correct or not in order to get more rewards. To solve this problem, penalties could be applied to such blockchain users for the purpose of removing malicious reviews; for example, a limitation could be set so that each blockchain user could only do a set number of reviews.The circumstances of rewards and penalties are shown in Sections 5.5 and 5.6.

5.4.1 Smart Contract Example

In this section, we present a simple example of a smart contract written in Solidity. The general structure of a smart contract is shown in Listing 5.1. It begins with the `pragma`line, which specifies which compiler version can execute the contract, and after that, the contract name is specified with the keyword `contract`. The contract permits variable and method definitions as well as instructor and destructor methods.

Listing 5.1: Smart contract structure

```
contract Contract_Name{
    // variables
    ...
    // Instructor
    // Destructor
    // Other methods
}
```

A descriptive smart contract is shown in Listing 5.2 for collecting funds from any node on the blockchain. The collected fund is transferred into the contract creator's wallet. Only the contract creator can destroy the contract,and the existing Ether will be transferred to the contract creator's wallet. The payable function `pay_for_us` is responsible for collecting funds from blockchain users and sending them to the contract creator's wallet.

Listing 5.2: Smart contract for collecting funds

```
pragma solidity ^0.4.9;
contract CollectingFund{
    address public contractCreator;
    // variable that stores blockchain address
    function CollectingFund(){ //constructor
        contractCreator = msg.sender;
    }
    function kill() { // destructor
        if (contractCreator == msg.sender) {
            selfdestruct(contractCreator);
        }
    }
    function pay_for_us(uint amount) payable{
    //The keyword payable is used when a function
    //accepts Ethers from blockchain users.
        contractCreator.transfer(amount);
    }
}
```

5.4.2 What is Behind a Smart Contract?

A blockchain is a trustless construction that does not require a third party to establish trust. It is a trusted platform, and we can use it for applications that require a trusted core to be used.The main idea of a smart contract is similar to a voting system. The situation begins once a new user requests a review of his or her MCQ answers. The MCQ answers can be sent directly to the blockchain users as arguments, or a URL to the MCQ answers can be sent. It can also be shown by logs supported by Solidity so that important information can be printed out.All the blockchain nodes can see when a new review is required. They start the reviewing process, and once they finish, they can choose a score value for the reviewfrom 0 to 1. Similar in appearance to a voting process, the review choices are the candidates, and the blockchain users are the voters, who can choose only one candidate (i.e., one choice). The reviewing process has a set time period. Once it is expired, the contract does not allow any more reviews and decides whether the user assessment was fair or not.

5.4.3 Blockchain-based Reviewing Systems for Cloud User Assessments

For the reviewing process, we present two systems. The first system enables reviews only from blockchain users who have previous experience with the service that was assessed by the user. The other system permits reviews from any blockchain node. The two systems are shown as follows:

1. Giving decisions only by a service-trusted community.

   We use the term 'service x–trusted community'to represent a set of blockchain users who have assessed the service, x, before in a fair way. We begin the trusted community set with the contract creator. We assume that the contract creator is trusted so that we can begin the reviewing process. Once a review process ends with a positive decision (i.e., that it is a fair assessment), the user who requested the review is now considered trustworthy and will be added to the trusted community set such that he or she can do future reviews of new review requests.

2. Giving decisions by any cloud user.

   Some individuals could interpret the previous method as a form of dictatorship-because only certain blockchain users are allowed to do reviews for every service. Thus, the second method removes this possibility of dictatorship by giving all blockchain users the ability to review any MCQ answers for any service assessment, creating a need to build trusted communities for the services.

5.5 Smart Contracts with a Trusted Community

For simplicity, we have enabled two review options for each blockchain reviewer: accept or reject. The contract is shown as follows:

- *The variables* are shown in Listing 5.3:
  - `contractCreator` stores the contract owner's address.
  - `trust_users` is a dynamic list of all trusted community blockchain users who can do the review.
  - `actual_reviewers` is a dynamic list of the actual blockchain users who did the review.

- `strange_user` represents the data for the MCQ answers that need to be reviewed. It also contains the details of the cloud user that submitted the assessment to be reviewed.
- `startTime` and `ReviewingTime` represent the starting time and the time slot for the reviewing process, respectively.
- `state` represents the current state of the contract, which is initialised for waiting.
- `choices`is a dynamic array of the choices made by each reviewer.

Listing 5.3: Smart contract variables

```
pragma solidity ^0.4.9;
contract ReviewingAssessments{
    address public contractCreator;
    reviewer [] public trust_users;
    reviewer [] public actual_reviewers;
    strange public new_user;
    uint public startTime;
    uint public ReviewingTime = 10;
    contractStates public state = contractStates.waiting;
    Choices [] public choices;
    ...
}
```

- The *contract states* are shown in Listing 5.4 by defining the enumeration `contractStates`. The `waiting` state means that the contract has no assessments to be reviewed and is waiting for a stranger to submit his or her assessment details, while the `reviewing` means that an assessment is being reviewed and the time slot has not expired. The `decision` state means that the reviewing slot time has expired, and the contract is calculating the final decision of whether to accept or reject the assessment (if it is fair or unfair, respectively). The last state is `rewarding` , which enables the contract to send rewards and apply penalties. The rewarding stage will be examined in detail later.

Listing 5.4: Smart contract states

```
enum contractStates{
```

```
    waiting,
    reviewing,
    decision,
    rewarding
}
```

- The `reviewerStates` enumeration is shown in Listing 5.5. The `Normal` state means that the blockchain user can do the review as well as receive a reward. The `warning` state indicates that the blockchain user is not being honest in his or her reviews; this blockchain user can still do reviews but will not receive a reward. At this stage, if the blockchain user does additional honest reviews, he or she can go back to the `normal` state. The last state, `suspended` , means that the user cannot do any further reviews until he or she has paid a fine in the form of a penalty applied to him or her due to his or her dishonesty in the reviewing process.

Listing 5.5: Different reviewer states

```
enum reviewerStates{
    Normal,
    Warning,
    Suspended
}
```

- The `reviewer, stranger` and `choices` structures: The `reviewer` contains the data for each blockchain user who belongs to a trusted community. It contains his or her account address, name, a Boolean value that represents whether the reviewer submitted his or her vote or not, the final decision of the current review process (defined by the `vote_index` variable), and whether he or she submitted his or her review or not (indicated by the `voted` variable). The details of the service assessment that need to be reviewed are stored by the structure `strange`. The review results – or the choices made – are defined by the structure `Choices` , where every choice has a choice name as well as an integer variable to record how many block chain users have selected this choice.

Listing 5.6: Structure definitions

```
struct reviewer{
    address  add;
```

```
        string name;
        bool voted;
        uint vote_index;
        reviewerStates ustate;
    }

    struct strange{
        address str_addr;
        string str_name;
        string questionnaire_url;
    }

    struct Choices{
    bytes32 choice_name;
    uint choice_count;
    }
```

- Mapping functions: Each blockchain user can be a potential reviewer. Hence, the mapping function shown in Listing 5.7 creates a list of reviewer structure instances for each possible address. The array name is `reviewers`.

Listing 5.7: Address -to- reviewer mapping

- Smart contract constructor: As shown in Listing 5.8, the function of the constructor is to store the owner's address, along with the owner's details, as the first member of the trusted community. It is also used to initialise the choices for the review results, with zero counting for each choice.

Listing 5.8: Smart contract instructor

```
function ReviewingAssessments(){
    contractCreator = msg.sender;
    trust_users.push(reviewer({add:contractCreator,
    name:"Creator",voted:false,vote_index:0,
    ustate:reviewerStates.Normal}));
    choices.push(Choices({choice_name:"Accept",
    choice_count:0}));
```

```
        choices.push(Choices({choice_name:"Reject",
        choice_count:0}));
    }
```

- Modifiers: The modifiers are useful for decreasing the maximum gas required to execute a smart contract. If there exists a method that can only be executed when a certain condition is satisfied, it is not efficient to use the normal method definition, and inside, we use a conditional check.This is because in all cases – whether the conditions are met or not – the maximum gas needed to execute the method must be paid. Modifiers are being used as a precondition before executing the method.If the modifier results in an exception, the method will not be executed, and, hence, the user will not have to pay for the gas it would have used. As shown in Listing 5.9, the modifier `isnotTrusted` is implemented to check whether a user with address d is not trusted, while the modifier `isTrusted` checks whether the user with address d is trusted. The modifiers `isnotSuspended` and `isSuspended` check whether the user with address d is not suspended or suspended, respectively. In addition, the modifier `didnotVote` is used to check if the user with address d did notsubmit his or her reviewing result. The last modifier is `timeout` , which used to check the expiration of the reviewing time slot when the contract is in the reviewing state.

Listing 5.9: Modifiers

```
    modifier isnotTrusted(address d){
        for(uint i=0; i<trust_users.length;i++){
            if(trust_users[i].add == d) throw;
        }
        _;
    }

    modifier isnotSuspended(address d){
        if(reviewers[d].ustate == reviewerStates.Suspended)
            throw;
        _;
    }

    modifier isSuspended(address d){
```

```
        if(reviewers[d].ustate != reviewerStates.Suspended)
            throw;
        _;
    }

    modifier isTrusted(address d){
        bool decision_status = false;
        for(uint i=0; i<trust_users.length;i++){
            if(trust_users[i].add == d){
                decision_status = true;
                break;
            }
        }
        if(decision_status == false) throw;
        _;
    }

    modifier didnotVote(address d){
        if(reviewers[d].voted == true ) throw;
        _;
    }

    modifier timeout(){
        if((now > startTime+ReviewingTime)&&(state !=
        contractStates.waiting)) _;
        else throw;
    }
```

- Review request for a service assessment: Once a cloud user finishes a service assessment, he or she sends a request to other blockchain users to review his or her MCQ answers. Then, once the request is triggered, the reviewing time slot begins to tick, while the contract state changes to the reviewing state. All the blockchain users can now see that there is a request that needs to be reviewed. This is shown in Listing 5.10 as the `RequestingReview` method, which can only be called by users who do not belong to the trusted community.

Listing 5.10: Review request for a new service assessment

```
function RequestingReview(string _name, string _url)
isnotTrusted(msg.sender){
    new_user.str_addr = msg.sender;
    new_user.str_name = _name;
    new_user.questionnaire_url = _url;
    state = contractStates.reviewing;
    startTime=now;
    state = contractStates.reviewing;
}
```

- Notifying reviewers: As shown in Listing 5.11, log events are used to notify the blockchain users (reviewers) about a new review request and about the overall decision once it is made.

Listing 5.11: Log events for notifying blockchain users

```
event LogReviewEnd(bool OverallDecision);
event LogReviewInitialized(
address contractCreator,
address new_user[str_addr],
string new_user[questionnaire_url],
uint votingTime);
```

- Submitting reviews: Once a blockchain user finishes his or her review, the `submit_review` method shown in Listing 5.12 can be used to choose a review result from the choices set previously in Listing 5.8. This method can be called only by blockchain users who belong to the trusted community. Those users must not have had their reviews submitted before and must not be in a suspended state.The review is accepted if and only if it is submitted within the reviewing time slot; otherwise, the method rejects this review and calls the `take_action` function, which begins deciding the overall action – whether accepting or rejecting the service assessment – to update the TR system, as suggested in previous chapters. Note that once a blockchain user chooses a result choice for his or her review, the choice counter is automatically increased by 1. So, by the end of the

reviewing time slot, we only need to check which choice has the highest count, and this is the final decision.

Listing 5.12: Submitting review results for a request

```
function submit_review(uint choice_index)
isTrusted(msg.sender)isnotSuspended(msg.sender)
didnotVote(msg.sender){
    if(now > (startTime+ReviewingTime)){
        take_action();
    }else{
        if((state != contractStates.reviewing)||
        reviewers[msg.sender].voted) throw;
        reviewers[msg.sender].voted = true;
        reviewers[msg.sender].vote_index=choice_index;
        choices[choice_index].choice_count++;
        actual_reviewers.push(reviewer({add:msg.sender,
        name:"", voted:true,vote_index:choice_index,
        ustate:reviewers[msg.sender].ustate}));
    }
}
```

- Taking actions after a time-out: Listing 5.13 shows the method `take_action` that is responsible for finding an overall decision for the request, based on the trusted community users' reviews. Then, a reward/penalty action is initiated; later, the contract will be reset to accept new review requests. If the overall decision for a review request is positive, the user who submitted that request will be added to the trusted community so that he or she can do reviews later. As a way of encouraging the blockchain users to review requests, we offer them rewards – for those who gave a review decision similar to the majority of decisions regarding the request – and apply penalties to those who gave a different review decision. A blockchain user can only receive rewards if that user is in the normal state. If a blockchain user in the normal state is subject to a penalty, his or her state will be changed to the warning state, which, of course, he or she can reverse by conducting better reviews, changing his or her state back to normal. If a blockchain user is in the warning state and he or she deserves a penalty, his or her state will be changed to suspended, which means that the user is not allowed to do further reviews without

paying a fee (in the form of Ether). Once a suspended user pays the fee through the function `clearRecord` , shown in Listing 5.14, his or her state goes back to normal.

Listing 5.13: Taking actions after a time-out

```
function take_action() timeout(){
    state = contractStates.decision;
    bool OverallDecision;
    if(choices[0].choice_count >= choices[1].choice_count){
        OverallDecision = true;
        trust_users.push(reviewer({add:new_user.str_addr,
        name:new_user.str_name,voted:false,vote_index:0,
        ustate:reviewerStates.Normal}));
    }else{
        OverallDecision = false;
    }
    state = contractStates.rewarding;
    for(uint i=0; i<actual_reviewers.length; i++){
        if((actual_reviewers[i].vote_index==0 &&
        OverallDecision==false)||
        (actual_reviewers[i].vote_index==1 &&
        OverallDecision==true)){// Rewarding
            if(reviewers[actual_reviewers[i].add].ustate==
            reviewerStates.Warning){
                reviewers[actual_reviewers[i].add].ustate=
                reviewerStates.Normal;
            }else{
                actual_reviewers[i].add.transfer(1);
            }

        }else{// Penality
            if (reviewers[actual_reviewers[i].add].ustate ==
            reviewerStates.Normal)
                reviewers[actual_reviewers[i].add].ustate=
                reviewerStates.Warning;
            else
```

```
                reviewers[actual_reviewers[i].add].ustate=
                reviewerStates.Suspended;
            }
            reviewers[actual_reviewers[i].add].voted=false;
        }
    // reset the contract state, choices count and
    the actual viewers list.
        state = contractStates.waiting;
        for(uint j=0; j<choices.length; j++)
            choices[j].choice_count=0;
        delete actual_reviewers;
    }
```

Listing 5.14: Paying fees for suspended users as a penalty

```
    public payable{
        contractCreator.transfer(1);
        reviewers[msg.sender].ustate = reviewerStates.Normal;
    }
```

5.6 Smart Contracts without a Trusted Community

The difference between the smart contract considered here and the one described in the previous section is that we do not need to construct a trusted community. Any blockchain user can do reviews if he or she is not suspended. Therefore, we do not need the `trust_users` dynamic list, and we no longer need to add members to this list. This can be done by removing the `trust_users.push` instruction from the constructor method. The modifiers `isTrusted` and `isnotTrusted` are also removed. We offer a simpler reward/penalty approach: We enable two reviews for every user state. This means that a user will receive rewards for his or her first two reviews but will not receive rewards for the second two reviews (i.e., they will be conducted for free). However, if the number of reviews done by a blockchain user exceeds four, the user state is changed to suspended, and the user must pay a fee to return to a normal state. We use a variable called `numberOfAssessments = 0` in the `reviewer` structure to keep track of how many reviews each user completes. As the `take_action` is different from the one described in the previous section, its new implementation is shown in Listing 5.15.

Listing 5.15: A modified take_action method

```
function take_action() timeout(){
    state = contractStates.decision;
    bool OverallDecision;
    if(choices[0].choice_count >= choices[1].choice_count){
        OverallDecision = true;
    }else{
        OverallDecision = false;
    }

    state = contractStates.rewarding;
    for(uint i=0; i<actual_reviewers.length; i++){
        if(reviewers[actual_reviewers[i].add].numberOfAssessments<=1 ){
            reviewers[actual_reviewers[i].add].numberOfAssessments++;
            actual_reviewers[i].add.transfer(1);
        }else{// warning and suspended
            reviewers[actual_reviewers[i].add].numberOfAssessments++;
        }
        reviewers[actual_reviewers[i].add].voted=false;
    }
    // reset the contract
    state = contractStates.waiting;
    for(uint j=0; j<choices.length; j++)
        choices[j].choice_count=0;
    delete actual_reviewers;
}
```

5.7 Summary

As blockchain technology is a trustless (or trust-free) platform, it does not require a third party to execute a transaction. In addition, it allows for decentralised decisions, in which no single user can influence the blockchain towards a specific decision. It guarantees secure operations over the network because of its use of the public cryptography method, which does not reveal private keys to other nodes. All of these features help us to redesign

our applications with an improved performance. We selected the Ethereum platform to build a filtering-out review system as a smart contract.Using the suggested method of reviewing smart contracts, we increased the reliability of our TR system, described in the previous chapters, by removing unfair assessments before updating the service trust values. The suggested smart contracts can be used, with minor changes to its rules, to describe any assessment or review type.

Chapter 6

Tests and Experiment Results

6.1 Introduction

In this chapter, we provide estimations, based on experiments, of an aggregation constant and an ageing factor that were made using multiple tests with human testers. Similar work was presented in a paper by (4). Additionally, we present, in Sections 6.2 and 6.3, descriptions of software we designed and settings for the tests included in this section. The test procedures provide favourable estimations for the aggregation constant and the ageing factor, shown in Sections 6.4 and 6.5.

6.2 Software

We implemented software, using Java and a Microsoft (MS) Access database, that provided a graphical user interface (GUI) that enabled both providers and cloud users to complete assessments. In this section, we present images of the tool interface and describe how it works. To begin, as shown in Figure 6.1, a tool login window supported three different logins: a cloud provider login, a cloud user login and a tester login, which was initially a cloud user login.

For cloud provider logins, we enabled the addition of new services to the system, as shown in Figure 6.2. We also performed provider self-assessments by allowing providers to answer a CAIQ, as shown in Figure 6.3. Note that the button that allowed each assessment to be submitted was disabled until each provider answered all CAIQ questions.

For cloud user logins, a user was asked to enter his or her details into the system and to choose a service to assess, as shown in Figure 6.4. Then, as shown in Figure 6.5, a user questionnaire appeared to the user that was to be answered. Note that for both

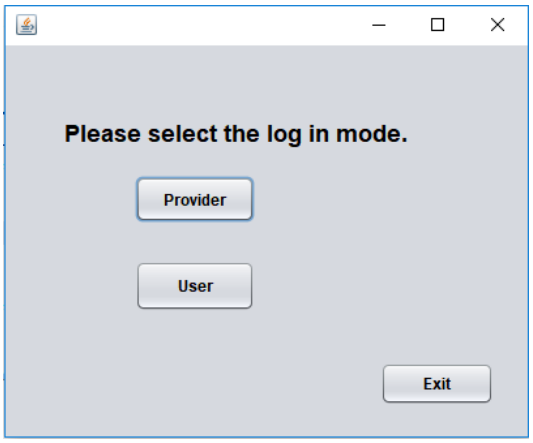

Figure 6.1: The tool's login window.

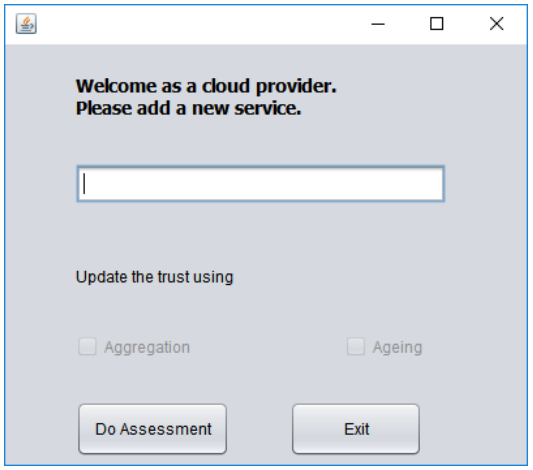

Figure 6.2: The tool's window for adding new services.

provider and user assessments, we allowed users and providers to give 'Yes', 'No' and 'Unknown' answers to MCQ questions.

Providers' and users' opinions were collected using the approach shown in Chapter 4, and the opinions were classified using the classifier in Section 4.3.2. For user assessments, an update action was performed either using only aggregation or using aggregation with ageing, as explained in Chapter 4. All assessment data, including questionnaire answers, providers' opinions, users' opinions and update effects, were stored in a database.

For tester logins, a tester logged in as a user and selected 'start random assessment' from the window shown in Figure 6.5. The tester assessment window is shown in Figure 6.6. The tester could set values for the aggregation constant and the ageing factor. The most recent trust value was then shown to the tester.

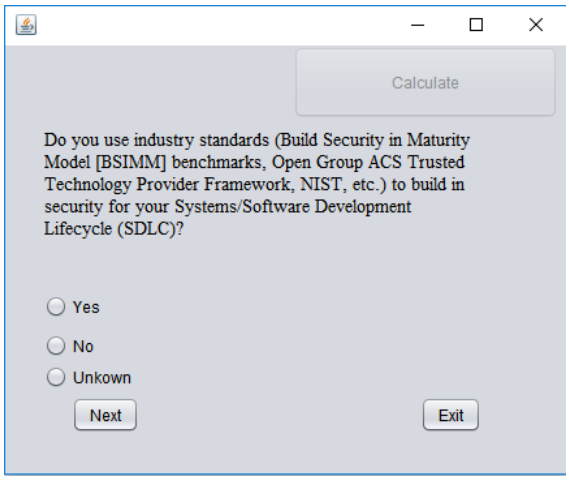

Figure 6.3: The tool's window for outputting self-assessment questionnaires.

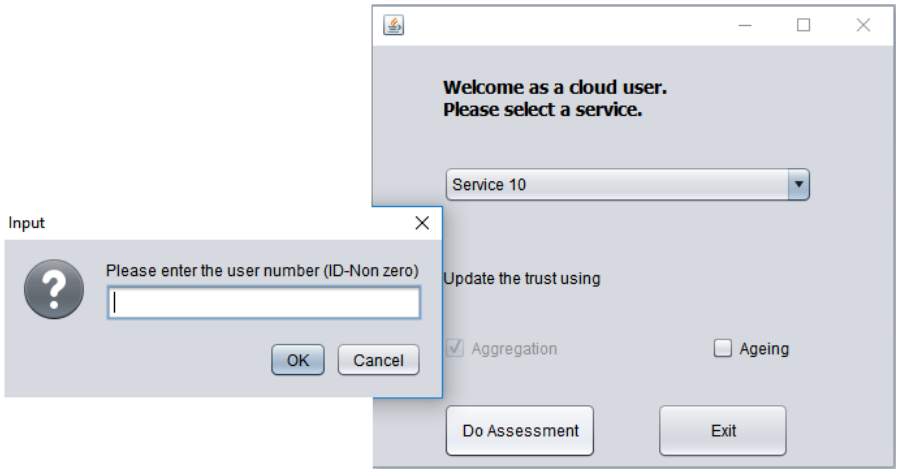

Figure 6.4: The tool's window for selecting services to be assessed by cloud users.

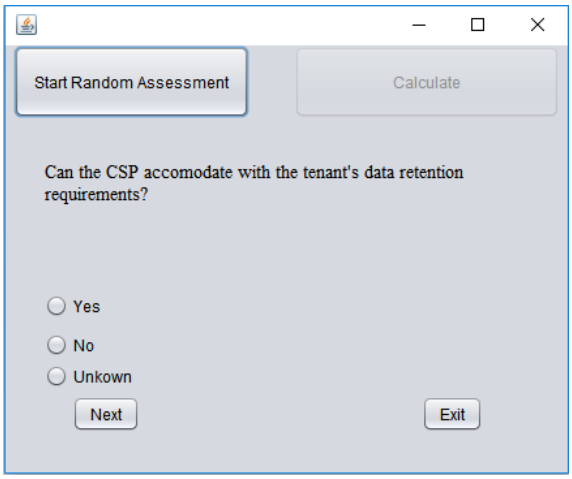

Figure 6.5: The tool's window for outputting cloud user questionnaires.

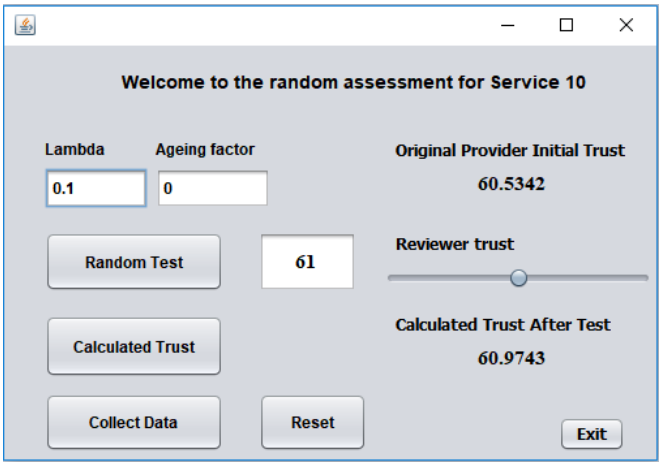

Figure 6.6: The tool's window for tester assessments.

Subsequently, the tester selected a Random Test button, and then he or she was asked, 'How many random opinions do you want to generate?' These opinions were shown to the tester. For example, Figures 6.7 and 6.9 were shown to the tester for testing situations with only aggregation and with aggregation and ageing, respectively.

Once the tester had an estimation for a new overall trust value, he or she could record this estimation in a text box and select it using a slider. A Calculated Trust button was used for the approach discussed in Chapter 4 in order to calculate a new overall trust value. The subsequent calculated value, as well as the estimated value, were collected using a Collect Data button.

6.3 Test Settings

Tester logins were designed to test this study's proposed assessment system using human testers. For such testing, we used randomly generated answers (values) for each user questionnaire. Once we had the random answers, we used the approach discussed in Chapter 4 to collect a random overall user opinion and visualise it inside a barycentric triangle (i.e., the triangle shown in Figure 4.1). A visualisation window was outputted to a user with rating classes, also shown in Figure 4.1, and the tester was asked to guess the overall trust value. The tester was informed of the most recent trust value, as well as the aggregation constant value, before giving his or her estimation. Note that the tester was not informed of the value of k for each rating class as we wanted to compare this study's proposed approach to humans' considerations. Then, the tester's estimation, as well as the actual trust value that was calculated using the approach in Section 4.3.2, were stored in a database. The tester repeated this procedure with different aggregation constants and different numbers denoting random users' opinions about a single service.

A set of experiments was conducted to obtain favourable selections for the aggregation and ageing factors, which were λ and Λ, respectively. This was done to achieve a minimum average error. The experiment steps are shown in Sections 6.4 and 6.5.

6.4 Aggregation Constant Estimations

In this section, the proposed method was used for multiple supervised tests in order to obtain favourable estimations of the aggregation constant λ that ranged from 0–1. By 'supervised', we mean that we had humans test our system. We used their reviews to readjust aggregation constants. The procedure is listed below.

1. Generate n random subjective opinions, $\omega_x^{\mathbb{A}} = \{\omega_x^{A_1}, \omega_x^{A_2}, ..., \omega_x^{A_n}\}$, that act as overall opinions generated from assessments completed by a set of users, $\mathbb{A} = \{A_1, A_2, ..., A_n\}$, for given service x.

2. Find a set of the opinions' ratings, $K = \{k_1, k_2, \ldots, k_n\}$.

3. Visualise the set of random opinions, $\omega_x^{\mathbb{A}}$, as shown in Figure 6.7.

4. Randomly choose an initial value for $\lambda \in [0, 1]$.

5. Ask a human tester to estimate the final trust value, $R^*_{x,(t+1)}$, after showing him or her the visualisation, the value of λ and the most recent trust value.

6. Let the program calculate the updated trust value using the following equation:

$$R_{x,(t+1)} = \sum_{i \leq n} k_i \times \lambda + R_{(x,t)}$$

,

where

$R_{x,(t+1)}$ is the updated trust value of the service x

and

$R_{x,t}$ is the latest trust value of service x before this process

.

7. Find an absolute error, $e = |R_{x,(t+1)} - R^*_{x,(t+1)}|$, and store the values λ and e in a database.

8. Repeat steps 1–7 with different n and λ values.

9. Repeat steps 1–8 with $m \geq 1$ testers (i.e., have other testers complete the entire process).

10. To collect more data, repeat this procedure for different $n \in \{1, 5, 10, 15, 20\}$ and $\lambda \in \{0.1, 0.2, 0.4, 0.5, 0.8, 1\}$ values.

11. For every λ, find an average absolute error:

$$\overline{e_\lambda} = \frac{\sum e_\lambda}{m \times n}$$

.

12. As the aim of these steps was to find the best value of λ (i.e., the value that was most similar to humans' estimations), select the best λ value that minimises the average absolute error.

For each test, the tool asked a tester to enter his or her details into the system; then, the testing procedure continued. For every tester, multiple tests were done with different λ and n values. The value of n represented how many random opinions were

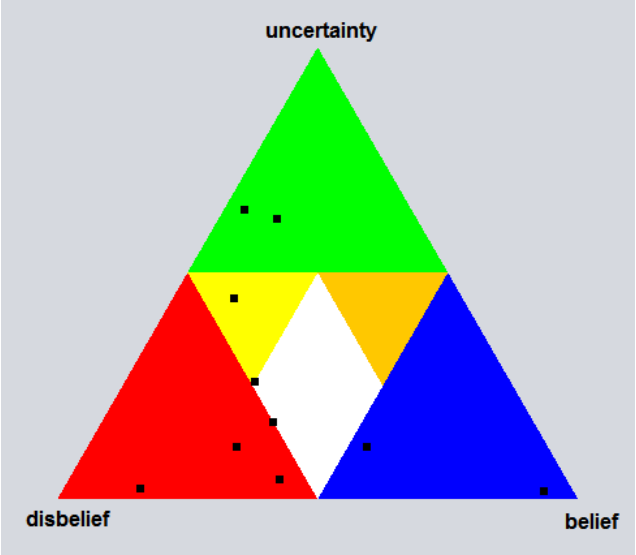

Figure 6.7: A visualisation of 10 random opinions.

outputted to the tester regarding a service. Once we showed visualised opinions to a user and informed him or her of the aggregation constant value and the most recent trust value of the service, the tester guessed a new trust value for the service while our program calculated the trust value using the aforementioned equations. The program then saved these values in a database. Subsequently, this process was repeated using all possible combinations of n and λ values.

In fact, this process was completed for $m = 10$ testers. The relationship between the average absolute error and the aggregation constant is shown in Figure 6.8. The horizontal axis shows values of $\lambda \in \{0.1, 0.2, 0.4, 0.5, 0.8, 1\}$. Additionally, the vertical axis shows average absolute errors $\overline{e_\lambda}$. We can conclude that for this study's proposed system, the best λ value that resulted in the minimum average error was 0.1. A simple justification of this value is that the services offered over clouds served millions of other cloud users. The updates needed for each assessment were very small because each assessment reflected only one opinion out of millions.

6.5 Ageing Factor Estimations

Similar to aggregation constants, to estimate ageing factors, we ran numerous supervised tests. However, we aimed to determine the best ageing factor value that resulted in a minimum average error. To elucidate, we used two operations in order to update trust values. The first one was an operation with an ageing factor and Λ, which affected the

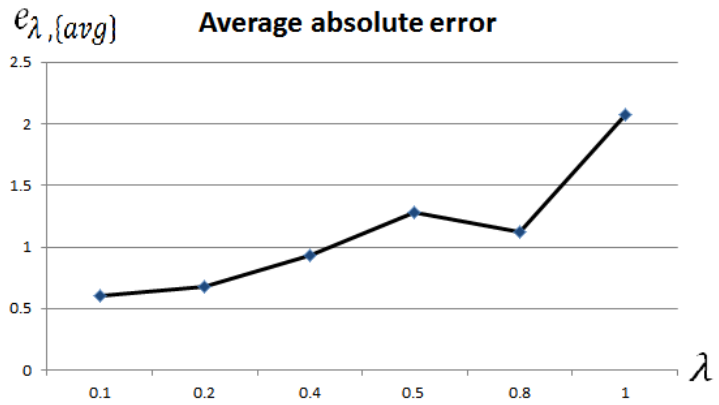

Figure 6.8: The relationship between the aggregation constant and the average absolute error, obtained after conducting 10 complete tests.

most recent trust value. The second one was an operation with the aggregation constant $\lambda = 0.1$ (this constant was determined in Section 6.4). The constant represented the update effect of a current user opinion. The procedure used for each test is listed below.

1. Assume that the aggregation constant is always $\lambda = 0.1$.
2. Complete the following steps for the testers $\{t_1, t_2, \ldots, t_m\}$; $m \in \mathbb{N}_{>0}$.
3. Additionally, complete the following steps for the various ageing factors $\Lambda \in \{0.01, 0.2, 0.4, 0.6, 0.8, 0.99\}$.
4. Ask each tester, t_i, to complete n random trials with every single Λ value.
5. For every trial, ask each tester to select the positive random integer l, which represents how many random opinions will be generated. Each opinion represents a different time frame.
6. Visualise each opinion individually, as shown in Figure 6.9. Additionally, ask each tester to give an estimated value for each updated trust value; this gives him or her the most recent trust value.
7. Let the program calculate the overall updated trust value, as described in Section 4.3.2.
8. Calculate the absolute error, e_Λ, between each estimated value and each calculated value.

9. Repeat steps 5–8 for the rest of the trials—that is, for $(l-1)$ trials.
10. Repeat steps 5–9 for all Λ values.
11. Repeat steps 5–10 with all other testers—that is, with $(m-1)$ testers.
12. After finishing the steps with all the testers, calculate, for every Λ, the average absolute error, $\overline{e_\Lambda} = \frac{\sum e_\Lambda}{l \times m}$.
13. Select the best ageing factor, Λ, with the minimum average absolute error $\overline{e_\Lambda}$.

For Figure 6.10, the test procedure comprised 10 testers. Each tester completed tests for $\Lambda = \{0.01, 0.2, 0.4, 0.6, 0.8, 0.99\}$. For each value, a tester completed 10 test situations, such that for trial i, we generated i random opinions at different time frames. For example, for trial $i = 1$, we generated only one opinion for a random user (it was the first time the user completed an assessment). Similarly, for trial $i = 2$, we generated two opinions for the random user, which means that the user had one previous assessment (the first one) in addition to the new assessment. Considering Figure 6.10, we can conclude that the best ageing factor with the minimum average absolute error was 0.01.

For our user system—cloud user assessments—we preferred low values for both the aggregation constant and the ageing factor. For the aggregation constant, we had a very large number of cloud users. Therefore, if we had a large aggregation constant, the trust value could change rapidly (i.e., it could rapidly increase or decrease after a small number of assessments, which is unfair, as the majority of users would not have completed assessments at that time. For the ageing factor, we preferred small values because for large Λ values, we gave cloud users the power to update trust values with sections larger than their rights. Consider the following.

Suppose that we have the aggregation constant λ and the ageing factor $\Lambda = 1$. Consider service y with 10 users. It is fair to give each user a small portion of his or her assessment history (e.g., an update of $\pm 0.1F$, where F is the maximum trust value) aggregated with the output of a new assessment by the user for the same service. During his or her first assessment, the user contributes an update of $\pm\lambda$, at most. For his or her second assessment, the user contributes $\pm\lambda$ with his or her opinion in addition to another $\pm\lambda$ from his or her previous opinion; the ageing factor is $\Lambda = 1$. Thus, with only the one user, the trust value can reach F after n assessments, without any consideration of the other 9 users' opinions.

If $\Lambda = 0.01$, the maximum update the user can do is $\pm(\lambda + 0.02\lambda) = 1.02\lambda$, which is acceptable, because for any user, his or her opinion of the service makes a contribution

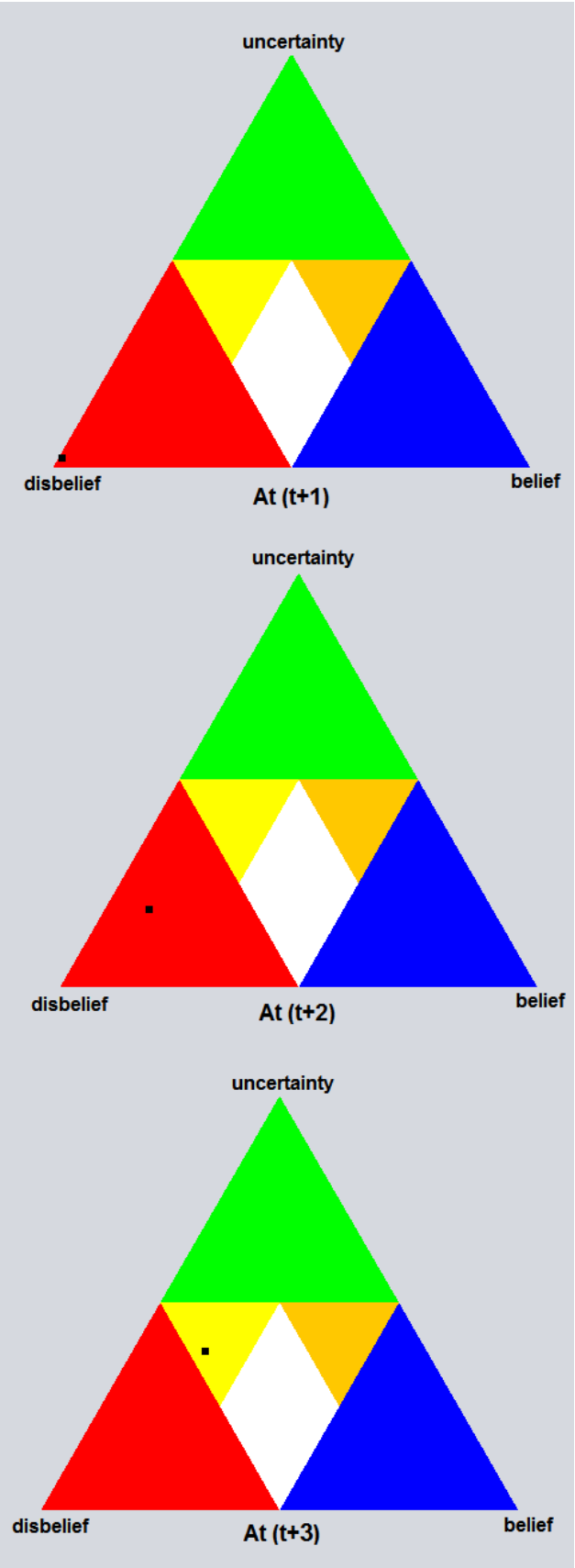

Figure 6.9: Three random opinions from the three consecutive time slots $(t+1)$, $(t+2)$ and $(t+3)$.

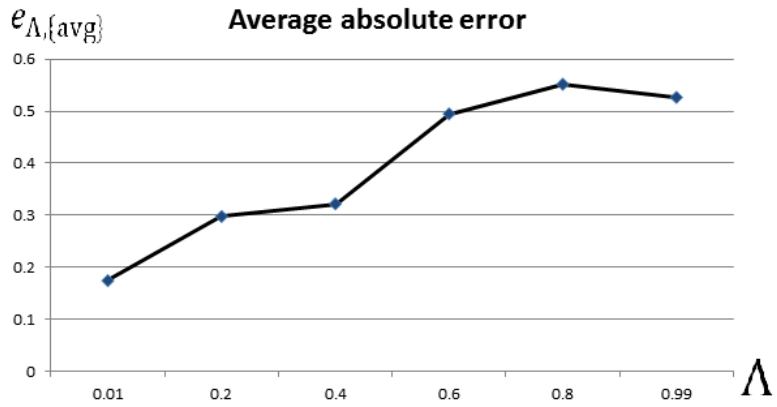

Figure 6.10: The relationship between the ageing factor and the average absolute error, obtained after conducting 10 complete tests.

of only $\pm 1.02\lambda$, which is close to a setting that occurs when a user gives only one opinion for a service that has a maximum update of $\pm\lambda$.

6.6 Measuring Barycentric Classifier Performance

In this section, we discuss a method for measuring the performance of the classifier mentioned in Section 4.3.2. The input for this classifier is a user's binomial opinion calculated using questionnaire answers, while the output for this classifier is the opinion's equivalent rating class. In order to calculate classifier performance, we conducted different tests. The test procedure was as follows.

First, we asked a group of 20 users—all the users had computer science backgrounds—to assess four different services by answering an MCQ provided during the user assessment process, which gave us a total of 80 assessment trials. The services were Dropbox, a cloud storage service; Gmail, an electronic mailing service created by Google; Hotmail, an electronic mailing service created by MSN and Facebook, a social network. We let each user answer the questionnaire, then asked him or her to select an equivalent rating class for his or her opinion. For each assessment trial, we used the method explained in Section 4.3.2 to calculate a rating class for each user's opinion, collected using the questionnaire. The classifier worked correctly if two rating classes were the same; otherwise, the classifier resulted in errors.

The aim of our experiment was to compare the rating classes calculated using the classifier with humans' expectations. This means that we were not interested in analysing

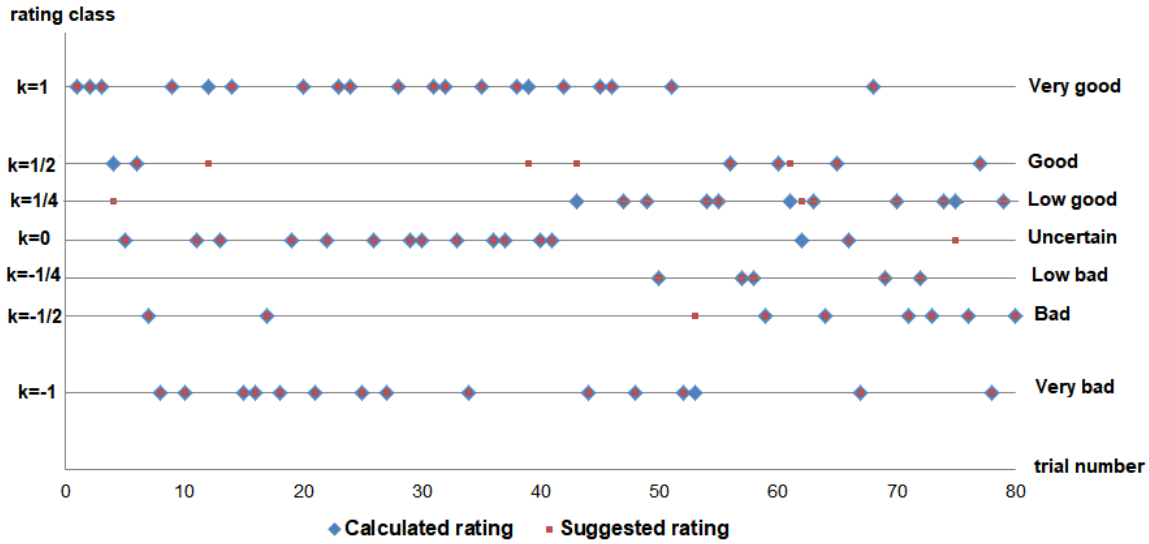

Figure 6.11: Tests of the barycentric classifier's performance.

every service individually. Thus, we combined the results for all the tests conducted for all the services in Figure 6.11. The figure shows the results of 80 tests. The horizontal axis shows the testing instances, while the vertical axis shows the opinion rating classes.

The barycentric classifier succeeded in generating a rating class the same as the suggested one, and it did so with an accuracy of 90%. The remaining 10% represented errors made by the classifier, and 1.7% of that percentage was the average absolute error percentage. For the errors, we noticed that the barycentric classifier produced a calculated rating, which was one step away (by a significant distance) from the suggested rating in both directions. We then developed a method to enhance the classification stage that generated an average of the calculated rating using the barycentric classifier and a rating suggested by a user. This method decreased the average absolute error percentage from 1.7% to 0.86%.

6.7 Summary

Considering the results of the experiments above, we can conclude that for our system, the aggregation constant value and ageing factor value need to be small. To clarify, 'small' is $\lambda = 0.1$ for the aggregation constant and $\Lambda = 0.01$ for the ageing factor. The reason for these small values is that when conducting aggregation with a high aggregation constant, trust values change rapidly as each step (i.e., each update that depends on

an aggregation constant) increases. As each user contributes with an update portion of each overall update that is applied to a service's original trust value, each user's update portion, at any time, should not exceed a certain value.

If the ageing factor value for a service is high when an update is completed using aggregation with ageing, the update portion of each user increases every time each user completes an additional assessment of the service. For example, when $\Lambda = 1$, only one user can make the service's overall trust value reach the top value of 100 or the bottom value of 0 because each assessment contributes with the user's current value in addition to the overall previous values generated by the user during past assessments. This can occur after a finite number of assessments of the service is completed by the user, which can create a problem in regards to the trust and reputation system.

Finally, we can conclude that the barycentric classifier performed its operations with an accuracy percentage of 90%. A rating class from the barycentric classifier can be used, as well as a suggested rating class from a user, to generate an average value. The value minimises the overall average absolute error to 0.86%.

Chapter 7

Discussion

7.1 Summary and Conclusions

This chapter summarises the contributions of this thesis and articulates some of the questions raised by this work. Our aim was to suggest ways to enhance the TR systems of cloud computing. Most TR systems are built on a one-sided opinion – that of the provider – based on self-assessments filled out by the provider about a service they offer.These assessments are done by answering the CAIQ, which is an MCQ provided by the Cloud Security Alliance (CSA). As any service provided over the cloud is offered to the cloud users, we note that it is better to design a TR system that reflects not only the provider's opinions but also the users' satisfaction level after they experience the service. Now, the trust values provided by the TR system we provide can be used as a reference as to whether a provider should enhance its service(s) or not.

In order to design a reputation system that reflected both the provider's opinions and the users' satisfaction level, we had to use a method that would enable the cloud user to express his or her satisfaction level. We found that the best approach was to allow the user to express his or her opinion by answering a questionnaire that assesses a service from different points of view, such as security, governance and access management. We used this assessment tool rather than giving the user the ability to enter one scalar value that would measure his or her level of satisfaction because individual measures are subjective; hence, two individuals with the same level of satisfaction could give two different scalar values. The problem we addressed was finding an MCQ that was compatible with the cloud user assessments. The CSA announced in the past that they would design a suitable questionnaire, but at present, it has not yet been released.

We found an assessment model that was created for expert users by the Smals 'ICT

for Society' group (44). This model enables a cloud user to assess the security levels of services offered over the cloud. Therefore, we used the Smals ICT for Society user assessment questionnaire to give the user the opportunity to express his or her feelings about cloud services, anticipating that the user's answers would reflect his or her satisfaction level.We enabled three answer options for each question: Yes, No, and Unknown. We enabled the Unknown option because it is better to receive an uncertain opinion than an incorrect one, as omitting this option would force the user to answer Yes or No to questions for which he or she may not have answers.

Neither the infrastructure we suggested to perform the assessments nor the use of the CTP protocol to get information about a service is complicated. The reason for this simplicityis that we used the elements we added, such as a GTS and an LTS, for different functions, depending on the task required. We did not make any contributions to the CTP protocol. We suggested that users could use the CTP protocol to request certain information about a specific service.

The primary work of this thesis consisted of the cloud user assessments, which began by giving each participating cloud user a copy of the assessment questionnaire until the digital trust value of the assessed service was updated.We faced many challenges while conducting these assessments. The first was determining how to extract the user satisfaction level from the MCQ answers as a value. The second challenge concerned cloud users that needed help with their assessments. We had to determine whether is was possible to ask another user to do a service assessment in place of the user who was originally supposed to do this task. The last and most important challenge was determining how we could make sure that each user assessment was done correctly and fairly. In other words, how could we detect and remove unfair assessments before updating the trust value?

To obtain the user satisfaction level from the questionnaire answers, we first extracted the users' opinions. Because of the uncertainty that could exist in the MCQ answers, the best way to describe the user opinion was by modelling it as a subjective opinion – specifically, a binomial subjective opinion. The next problem was extracting the satisfaction level from a user's binomial opinion. In order to do that, a barycentric classifier was proposed to classify any binomial opinion into one of six rating classes, which could be defined later as a scalar value, k. We introduced two ways in which the value of the rating class, k, could update the latest trust value: by using aggregation only or by using aggregation with ageing, which includes the assessment history of the service.

We also offered some assistance for users during the assessment procedure. We suggested that a cloud user could ask an expert cloud user to do the assessment for him or her if the cloud user trusted that expert.This feature was developed using a fusion transitivity operator to transfer opinions between cloud users. We also gave these users the ability to define the scaling ratio of each assessment domain while calculating his or her overall opinion. This was accomplished using weighted subjective operators, such as consensus and averaging.

We provided an implementation for the suggested approaches using Java. We used this implementation to provide estimated values for the aggregation constant and the ageing factor. According to the results of the experiments, it is better to have small values for both of these. In our settings, these values were $\lambda = 0.1$ for the aggregation constant and $\Lambda = 0.01$ for the ageing factor. A limitation here was that we provided an estimation based on experiments that were run with a small number of users who had a computer science background. Furthermore, the tests were run to assess four cloud services. In the future, we intend to ask a wider variety of users with different backgrounds to participate in experiments on a greater number of services.

We tested the technique suggested in Section 4.3.2 that generates a user's opinion from his or her MCQ answers and classifies it into a rating class. The test results indicate that this technique assigned a rating class that was similar to the rating suggested by the user, with 90% accuracy and an average absolute error rate of 1.7%.The average error ratio could be decreased to 0.86% if we were to adjust our approach to yield the average of the calculated rating as well as that of the suggested rating.This modification would provide better performance as the opinion rating class is generated from an assessment questionnaire for the service, which would model the service operation correctly.We could also use the overall suggested rating value inputted by the user to decrease this error.

The final contribution of this research is filtering out unfair opinions before they can affect the TR system. We thought that it would be best to search for a trusted platform on which we could implement the filtering approach. We used Ethereum, which is based on blockchain technology and is a trustless platform.Operations can be done without a third party on this decentralised platform, where no single person or group of people can influence the decisions of the filtering method. We implemented this idea in the form of a smart contract, which is self-executing, without any external control. The aim was to give the blockchain userthe ability to see the answers of other cloud users to the assessment questionnaire. We then asked the blockchain users to review each assessment answer and submit the result, whether by accepting it as a fair assessment or

by rejecting it. The problem we faced at this point was how we could buy cryptocurrency (virtual tokens), such as Ether or Bitcoin, so that we could publish the contract over the blockchain and then test it. We solved this problem by testing the implemented smart contracts over a test network supported by Ethereum instead of over the main network. For the test network, we used cryptocurrency to execute transactions instead of real money.By changing the contractual rules, the suggested smart contract can be used in different applications that require a form of voting or rating from blockchain users.

We implemented two different contracts to execute the filtering operation of unfair assessments. The main difference between them is the blockchain users who are involved in the reviewing process. The first contract we presented in Section 5.5 constrains the reviewing of any user service assessment to be done only by the users belonging to the service's trusted community. The reason behind this constraint is that for a blockchain user to be able to review another user's service assessment, he or she must have experience using that service and must understand its features, strengths and weaknesses in order to determine whether or not the submitted user assessment was done fairly.From this contract, however, we thought that a dictatorship could develop in the cloud community because only the users who do the reviewing can belong to a trusted service community.A solution for the dictatorship problem was outlined in the second contract, which was presented in Section 5.6. In this contract, we gave any blockchain user the ability to review any assessment request so that no single group would control the decisions regarding a specific service. We also suggested a reward/penalty system that rewards the blockchain users involved in the reviewing process and applies penalties to those who try to do more reviews.

As a summary of our contributions, we provided a way for cloud users to do assessments in order to have their opinions reflected in the trust values generated by a TR system. To accomplish this aim, we proposed a barycentric classifier that categorises any binomial subjective opinion into a rating class (one of six classes). We also discussed how a cloud user can ask expert users to do his or her assessment by using opinion transitivity. Moreover, we showed how we can use weighted fusion operators to give a user the ability to define his or her own assessment scheme and assign weights to every aspect of the assessment, reflecting his or her needs. Finally, we used blockchain technology to facilitate a reviewing process for all the assessments done by cloud users before updating the TR system, on behalf of the blockchain users.

7.2 Future Work

We intend to test the system using a large number of human testers that have different backgrounds (beyond a computer science background) over numerous cloud services. We aim to implement the suggested smart contract using EthereumJ so that we can develop a stand-alone tool for cloud user assessments. Another aim is to secure the tool against any malicious attacks that could affect the trust values generated by the suggested TR system. We want to design a new classifier based on artificial intelligence to enhance the opinion classification performance, which was done in this work using the proposed barycentric classifier.

Appendix A

Cloud Provider Questionnaire

Here we show the questions used in the cloud provider questionnaire.These questions can be answered by Yes, No or Unknown choices (18).

1. Do you use industry standards (Build Security in Maturity Model [BSIMM] benchmarks, Open Group ACS Trusted Technology Provider Framework, NIST, etc.) to build in security for your Systems/Software Development Lifecycle (SDLC)?

2. Do you use an automated source code analysis tool to detect security defects in code prior to production?

3. Do you use manual source-code analysis to detect security defects in code prior to production?

4. Do you verify that all of your software suppliers adhere to industry standards for Systems/Software Development Lifecycle (SDLC) security? (SaaS only) Do you review your applications for security vulnerabilities and address any issues prior to deployment to production?

5. Are all identified security, contractual and regulatory requirements for customer access contractually addressed and remediated prior to granting customers access to data, assets and information systems?

6. Are all requirements and trust levels for customers' access defined and documented?

7. Are data input and output integrity routines (i.e., reconciliation and edit checks) implemented for application interfaces and databases to prevent manual or systematic processing errors or corruption of data?

8. Is your Data Security Architecture designed using an industry standard (e.g., CDSA, MULITSAFE, CSA Trusted Cloud Architectural Standard, FedRAMP, CAESARS)?

9. Do you produce audit assertions using a structured, industry accepted format (e.g., CloudAudit/A6 URI Ontology, CloudTrust, SCAP/CYBEX, GRC XML, ISACA's Cloud Computing Management Audit/Assurance Program, etc.)?

10. Do you allow tenants to view your SOC2/ISO 27001 or similar third-party audit or certification reports?

11. Do you conduct network penetration tests of your cloud service infrastructure regularly as prescribed by industry best practices and guidance?

12. Do you conduct application penetration tests of your cloud infrastructure regularly as prescribed by industry best practices and guidance?

13. Do you conduct internal audits regularly as prescribed by industry best practices and guidance?

14. Do you conduct external audits regularly as prescribed by industry best practices and guidance?

15. Are the results of the penetration tests available to tenants at their request?

16. Are the results of internal and external audits available to tenants at their request?

17. Do you have an internal audit program that allows for cross-functional audit of assessments?

18. Do you have the ability to logically segment or encrypt customer data such that data may be produced for a single tenant only, without inadvertently accessing another tenant's data?

19. Do you have capability to recover data for a specific customer in the case of a failure or data loss?

20. Do you have the capability to restrict the storage of customer data to specific countries or geographic locations?

21. Do you have a program in place that includes the ability to monitor changes to the regulatory requirements in relevant jurisdictions, adjust your security program for changes to legal requirements, and ensure compliance with relevant regulatory requirements?

22. Do you provide tenants with geographically resilient hosting options?

23. Do you provide tenants with infrastructure service failover capability to other providers?

24. Are business continuity plans subject to test at planned intervals or upon significant organizational or environmental changes to ensure continuing effectiveness?

25. Do you provide tenants with documentation showing the transport route of their data between your systems?

26. Can tenants define how their data is transported and through which legal jurisdictions?

27. Are information system documents (e.g., administrator and user guides, architecture diagrams, etc.) made available to authorized personnel to ensure configuration, installation and operation of the information system?

28. Is physical protection against damage (e.g., natural causes, natural disasters, deliberate attacks) anticipated and designed with countermeasures applied?

29. Are any of your data centers located in places that have a high probability/occurrence of high-impact environmental risks (floods, tornadoes, earthquakes, hurricanes, etc.)?

30. If using virtual infrastructure, does your cloud solution include independent hardware restore and recovery capabilities?

31. If using virtual infrastructure, do you provide tenants with a capability to restore a Virtual Machine to a previous state in time?

32. If using virtual infrastructure, do you allow virtual machine images to be downloaded and ported to a new cloud provider?

33. If using virtual infrastructure, are machine images made available to the customer in a way that would allow the customer to replicate those images in their own off-site storage location?

34. Does your cloud solution include software/provider independent restore and recovery capabilities?

35. Are security mechanisms and redundancies implemented to protect equipment from utility service outages (e.g., power failures, network disruptions, etc.)?

36. Do you provide tenants with ongoing visibility and reporting of your operational Service Level Agreement (SLA) performance?

37. Do you make standards-based information security metrics (CSA, CAMM, etc.) available to your tenants?

38. Do you provide customers with ongoing visibility and reporting of your SLA performance?

39. Are policies and procedures established and made available for all personnel to adequately support services operations' roles?

40. Do you have technical control capabilities to enforce tenant data retention policies?

41. Do you have a documented procedure for responding to requests for tenant data from governments or third parties?

42. Have you implemented backup or redundancy mechanisms to ensure compliance with regulatory, statutory, contractual or business requirements?

43. Do you test your backup or redundancy mechanisms at least annually?

44. Are policies and procedures established for management authorization for development or acquisition of new applications, systems, databases, infrastructure, services, operations and facilities?

45. Is documentation available that describes the installation, configuration and use of products/services/features?

46. Do you have controls in place to ensure that standards of quality are being met for all software development?

47. Do you have controls in place to detect source code security defects for any outsourced software development activities?

48. Do you provide your tenants with documentation that describes your quality assurance process?

49. Is documentation describing known issues with certain products/services available?

50. Are there policies and procedures in place to triage and remedy reported bugs and security vulnerabilities for product and service offerings?

51. Are mechanisms in place to ensure that all debugging and test code elements are removed from released software versions?

52. Do you have controls in place to restrict and monitor the installation of unauthorized software onto your systems?

53. Do you provide tenants with documentation that describes your production change management procedures and their roles/rights/responsibilities within it?

54. Do you provide a capability to identify virtual machines via policy tags/metadata (e.g., tags can be used to limit guest operating systems from booting/instantiating/transporting data in the wrong country)?

55. Do you provide a capability to identify hardware via policy tags/metadata/hardware tags (e.g., TXT/TPM, VN-Tag, etc.)?

56. Do you have a capability to use system geographic location as an authentication factor?

57. Can you provide the physical location/geography of storage of a tenant's data upon request?

58. Can you provide the physical location/geography of storage of a tenant's data in advance?

59. Do you follow a structured data-labeling standard (e.g., ISO 15489, Oasis XML Catalog Specification, CSA data type guidance)?

60. Do you allow tenants to define acceptable geographical locations for data routing or resource instantiation?

61. Do you inventory, document, and maintain data flows for data that is resident (permanent or temporary) within the services' applications and infrastructure network and systems?

62. Can you ensure that data does not migrate beyond a defined geographical residency?

63. Do you provide open encryption methodologies (3.4ES, AES, etc.) to tenants in order for them to protect their data if it is required to move through public networks (e.g., the Internet)?

64. Do you utilize open encryption methodologies any time your infrastructure components need to communicate with each other via public networks (e.g., Internet-based replication of data from one environment to another)?

65. Are policies and procedures established for labeling, handling and the security of data and objects that contain data?

66. Are mechanisms for label inheritance implemented for objects that act as aggregate containers for data?

67. Do you have procedures in place to ensure production data shall not be replicated or used in non-production environments?

68. Are the responsibilities regarding data stewardship defined, assigned, documented and communicated?

69. Do you support secure deletion (e.g., degaussing/cryptographic wiping) of archived and backed-up data as determined by the tenant?

70. Can you provide a published procedure for exiting the service arrangement, including assurance to sanitize all computing resources of tenant data once a customer has exited your environment or has vacated a resource?

71. Do you maintain a complete inventory of all of your critical assets that includes ownership of the asset?

72. Do you maintain a complete inventory of all of your critical supplier relationships?

73. Are physical security perimeters (e.g., fences, walls, barriers, guards, gates, electronic surveillance, physical authentication mechanisms, reception desks and security patrols) implemented?

74. Is automated equipment identification used as a method to validate connection authentication integrity based on known equipment location?

75. Do you provide tenants with documentation that describes scenarios in which data may be moved from one physical location to another? (e.g., offsite backups, business continuity failovers, replication)

76. Can you provide tenants with evidence documenting your policies and procedures governing asset management and repurposing of equipment?

77. Can you provide evidence that policies, standards and procedures have been established for maintaining a safe and secure working environment in offices, rooms, facilities and secure areas?

78. Can you provide evidence that your personnel and involved third parties have been trained regarding your documented policies, standards and procedures?

79. Do you allow tenants to specify which of your geographic locations their data is allowed to move into/out of (to address legal jurisdictional considerations based on where data is stored vs. accessed)?

80. Are ingress and egress points, such as service areas and other points where unauthorized personnel may enter the premises, monitored, controlled and isolated from data storage and process?

81. Do you restrict physical access to information assets and functions by users and support personnel?

82. Do you have key management policies binding keys to identifiable owners?

83. Do you have a capability to allow creation of unique encryption keys per tenant?

84. Do you have a capability to manage encryption keys on behalf of tenants?

85. Do you maintain key management procedures?

86. Do you have documented ownership for each stage of the lifecycle of encryption keys?

87. Do you utilize any third party/open source/proprietary frameworks to manage encryption keys?

88. Do you encrypt tenant data at rest (on disk/storage) within your environment?

89. Do you leverage encryption to protect data and virtual machine images during transport across and between networks and hypervisor instances?

90. Do you support tenant-generated encryption keys or permit tenants to encrypt data to an identity without access to a public key certificate (e.g., identity-based encryption)?

91. Do you have documentation establishing and defining your encryption management policies, procedures and guidelines?

92. Do you have platform and data appropriate encryption that uses open/validated formats and standard algorithms?

93. Are your encryption keys maintained by the cloud consumer or a trusted key management provider?

94. Do you store encryption keys in the cloud?

95. Do you have separate key management and key usage duties?

96. Do you have documented information security baselines for every component of your infrastructure (e.g., hypervisors, operating systems, routers, DNS servers, etc.)?

97. Do you have a capability to continuously monitor and report the compliance of your infrastructure against your information security baselines?

98. Do you allow your clients to provide their own trusted virtual machine image to ensure conformance to their own internal standards?

99. Do you provide security control health data in order to allow tenants to implement industry standard Continuous Monitoring (which allows continual tenant validation of your physical and logical control status)?

100. Do you conduct risk assessments associated with data governance requirements at least once a year?

101. Are your technical, business, and executive managers responsible for maintaining awareness of and compliance with security policies, procedures, and standards for both themselves and their employees as they pertain to the manager and employees' area of responsibility?

102. Do you provide tenants with documentation describing your Information Security Management Program (ISMP)?

103. Do you review your Information Security Management Program (ISMP) least once a year?

104. Do you ensure your providers adhere to your information security and privacy policies?

105. Do your information security and privacy policies align with industry standards (ISO-27001, ISO-22307, CoBIT, etc.)?

106. Do you have agreements to ensure your providers adhere to your information security and privacy policies?

107. Can you provide evidence of due diligence mapping of your controls, architecture and processes to regulations and/or standards?

108. Do you disclose which controls, standards, certifications and/or regulations you comply with?

109. Is a formal disciplinary or sanction policy established for employees who have violated security policies and procedures?

110. Are employees made aware of what actions could be taken in the event of a violation via their policies and procedures?

111. Do risk assessment results include updates to security policies, procedures, standards and controls to ensure they remain relevant and effective?

112. Do you notify your tenants when you make material changes to your information security and/or privacy policies?

113. Do you perform, at minimum, annual reviews to your privacy and security policies?

114. Are formal risk assessments aligned with the enterprise-wide framework and performed at least annually, or at planned intervals, determining the likelihood and impact of all identified risks, using qualitative and quantitative methods?

115. Is the likelihood and impact associated with inherent and residual risk determined independently, considering all risk categories (e.g., audit results, threat and vulnerability analysis, and regulatory compliance)?

116. Do you have a documented, organization-wide program in place to manage risk?

117. Do you make available documentation of your organization-wide risk management program?

118. Are systems in place to monitor for privacy breaches and notify tenants expeditiously if a privacy event may have impacted their data?

119. Is your Privacy Policy aligned with industry standards? Pursuant to local laws, regulations, ethics and contractual constraints, are all employment candidates, contractors and involved third parties subject to background verification?

120. Do you specifically train your employees regarding their specific role and the information security controls they must fulfill?

121. Do you document employee acknowledgment of training they have completed?

122. Are all personnel required to sign NDA or Confidentiality Agreements as a condition of employment to protect customer/tenant information?

123. Is successful and timed completion of the training program considered a prerequisite for acquiring and maintaining access to sensitive systems?

124. Are personnel trained and provided with awareness programs at least once a year?

125. Are documented policies, procedures and guidelines in place to govern change in employment and/or termination?

126. Do the above procedures and guidelines account for timely revocation of access and return of assets?

127. Are policies and procedures established and measures implemented to strictly limit access to your sensitive data and tenant data from portable and mobile devices

(e.g., laptops, cell phones and personal digital assistants (PDAs)), which are generally higher-risk than non-portable devices (e.g., desktop computers at the provider organization's facilities)?

128. Are requirements for non-disclosure or confidentiality agreements reflecting the organization's needs for the protection of data and operational details identified, documented and reviewed at planned intervals?

129. Do you provide tenants with a role definition document clarifying your administrative responsibilities versus those of the tenant?

130. Do you provide documentation regarding how you may or access tenant data and metadata?

131. Do you collect or create metadata about tenant data usage through inspection technologies (search engines, etc.)?

132. Do you allow tenants to opt out of having their data/metadata accessed via inspection technologies?

133. Do you provide a formal, role-based, security awareness training program for cloud-related access and data management issues (e.g., multi-tenancy, nationality, cloud delivery model segregation of duties implications and conflicts of interest) for all persons with access to tenant data?

134. Are administrators and data stewards properly educated on their legal responsibilities with regard to security and data integrity?

135. Are users made aware of their responsibilities for maintaining awareness and compliance with published security policies, procedures, standards and applicable regulatory requirements?

136. Are users made aware of their responsibilities for maintaining a safe and secure working environment?

137. Are users made aware of their responsibilities for leaving unattended equipment in a secure manner?

138. Do your data management policies and procedures address tenant and service level conflicts of interests?

139. Do your data management policies and procedures include a tamper audit or software integrity function for unauthorized access to tenant data?

140. Does the virtual machine management infrastructure include a tamper audit or software integrity function to detect changes to the build/configuration of the virtual machine?

141. Do you restrict, log and monitor access to your information security management systems? (E.g., hypervisors, firewalls, vulnerability scanners, network sniffers, APIs, etc.)

142. Do you monitor and log privileged access (administrator level) to information security management systems?

143. Do you have controls in place ensuring timely removal of systems access that is no longer required for business purposes?

144. Do you provide metrics to track the speed with which you are able to remove systems access that is no longer required for business purposes?

145. Do you use dedicated secure networks to provide management access to your cloud service infrastructure?

146. Do you manage and store the identity of all personnel who have access to the IT infrastructure, including their level of access?

147. Do you manage and store the user identity of all personnel who have network access, including their level of access?

148. Do you provide tenants with documentation on how you maintain segregation of duties within your cloud service offering?

149. Are controls in place to prevent unauthorized access to your application, program or object source code, and assure it is restricted to authorized personnel only?

150. Are controls in place to prevent unauthorized access to tenant application, program or object source code, and assure it is restricted to authorized personnel only?

151. Do you provide multi-failure disaster recovery capability?

152. Do you monitor service continuity with upstream providers in the event of provider failure?

153. Do you have more than one provider for each service you depend on?

154. Do you provide access to operational redundancy and continuity summaries, including the services you depend on?

155. Do you provide the tenant the ability to declare a disaster?

156. Do you provided a tenant-triggered failover option?

157. Do you share your business continuity and redundancy plans with your tenants?

158. Do you document how you grant and approve access to tenant data?

159. Do you have a method of aligning provider and tenant data classification methodologies for access control purposes?

160. Does your management provision the authorization and restrictions for user access (e.g., employees, contractors, customers (tenants), business partners and/or suppliers) prior to their access to data and any owned or managed (physical and virtual) applications, infrastructure systems and network components?

161. Do your provide upon request user access (e.g., employees, contractors, customers (tenants), business partners and/or suppliers) to data and any owned or managed (physical and virtual) applications, infrastructure systems and network components?

162. Do you require at least annual certification of entitlements for all system users and administrators (exclusive of users maintained by your tenants)?

163. If users are found to have inappropriate entitlements, are all remediation and certification actions recorded?

164. Will you share user entitlement remediation and certification reports with your tenants, if inappropriate access may have been allowed to tenant data?

165. Is timely deprovisioning, revocation or modification of user access to the organizations systems, information assets and data implemented upon any change in status of employees, contractors, customers, business partners or involved third parties?

166. Is any change in user access status intended to include termination of employment, contract or agreement, change of employment or transfer within the organization?

167. Do you support use of, or integration with, existing customer-based Single Sign On (SSO) solutions to your service?

168. Do you use open standards to delegate authentication capabilities to your tenants?

169. Do you support identity federation standards (SAML, SPML, WS-Federation, etc.) as a means of authenticating/authorizing users?

170. Do you have a Policy Enforcement Point capability (e.g., XACML) to enforce regional legal and policy constraints on user access?

171. Do you have an identity management system (enabling classification of data for a tenant) in place to enable both role-based and context-based entitlement to data?

172. Do you provide tenants with strong (multifactor) authentication options (digital certs, tokens, biometrics, etc.) for user access?

173. Do you allow tenants to use third-party identity assurance services?

174. Do you support password (minimum length, age, history, complexity) and account lockout (lockout threshold, lockout duration) policy enforcement?

175. Do you allow tenants/customers to define password and account lockout policies for their accounts?

176. Do you support the ability to force password changes upon first logon?

177. Do you have mechanisms in place for unlocking accounts that have been locked out (e.g., self-service via email, defined challenge questions, manual unlock)?

178. Are utilities that can significantly manage virtualized partitions (e.g., shutdown, clone, etc.) appropriately restricted and monitored?

179. Do you have a capability to detect attacks that target the virtual infrastructure directly (e.g., shimming, Blue Pill, Hyper jumping, etc.)?

180. Are attacks that target the virtual infrastructure prevented with technical controls?

181. Are file integrity (host) and network intrusion detection (IDS) tools implemented to help facilitate timely detection, investigation by root cause analysis and response to incidents?

182. Is physical and logical user access to audit logs restricted to authorized personnel?

183. Can you provide evidence that due diligence mapping of regulations and standards to your controls/architecture/processes has been done?

184. Are audit logs centrally stored and retained?

185. Are audit logs reviewed on a regular basis for security events (e.g., with automated tools)?

186. Do you log and alert any changes made to virtual machine images regardless of their running state (e.g., dormant, off or running)?

187. Are changes made to virtual machines, or moving of an image and subsequent validation of the image's integrity, made immediately available to customers through electronic methods (e.g., portals or alerts)?

188. Do you use a synchronized time-service protocol (e.g., NTP) to ensure all systems have a common time reference?

189. Do you provide documentation regarding what levels of system (network, storage, memory, I/O, etc.) oversubscription you maintain and under what circumstances/scenarios?

190. Do you restrict use of the memory oversubscription capabilities present in the hypervisor?

191. Do your system capacity requirements take into account current, projected and anticipated capacity needs for all systems used to provide services to the tenants?

192. Is system performance monitored and tuned in order to continuously meet regulatory, contractual and business requirements for all the systems used to provide services to the tenants?

193. Do security vulnerability assessment tools or services accommodate the virtualization technologies being used (e.g., virtualization aware)?

194. For your IaaS offering, do you provide customers with guidance on how to create a layered security architecture equivalence using your virtualized solution?

195. Do you regularly update network architecture diagrams that include data flows between security domains/zones?

196. Do you regularly review for appropriateness the allowed access/connectivity (e.g., firewall rules) between security domains/zones within the network?

197. Are all firewall access control lists documented with business justification?

198. Are operating systems hardened to provide only the necessary ports, protocols and services to meet business needs using technical controls (i.e. antivirus, file integrity monitoring and logging) as part of their baseline build standard or template?

199. For your SaaS or PaaS offering, do you provide tenants with separate environments for production and test processes?

200. For your IaaS offering, do you provide tenants with guidance on how to create suitable production and test environments?

201. Do you logically and physically segregate production and non-production environments?

202. Are system and network environments protected by a firewall or virtual firewall to ensure business and customer security requirements?

203. Are system and network environments protected by a firewall or virtual firewall to ensure compliance with legislative, regulatory and contractual requirements?

204. Are system and network environments protected by a firewall or virtual firewall to ensure separation of production and non-production environments?

205. Are system and network environments protected by a firewall or virtual firewall to ensure protection and isolation of sensitive data?

206. Are secured and encrypted communication channels used when migrating physical servers, applications or data to virtual servers?

207. Do you use a network segregated from production-level networks when migrating physical servers, applications or data to virtual servers?

208. Do you restrict personnel access to all hypervisor management functions or administrative consoles for systems hosting virtualized systems based on the principle of least privilege and supported through technical controls (e.g., two-factor authentication, audit trails, IP address filtering, firewalls and TLS-encapsulated communications to the administrative consoles)?

209. Are policies and procedures established and mechanisms configured and implemented to protect the wireless network environment perimeter and to restrict unauthorized wireless traffic?

210. Are policies and procedures established and mechanisms implemented to ensure wireless security settings are enabled with strong encryption for authentication and transmission, replacing vendor default settings? (e.g., encryption keys, passwords, SNMP community strings)

211. Are policies and procedures established and mechanisms implemented to protect wireless network environments and detect the presence of unauthorized (rogue) network devices for a timely disconnect from the network?

212. Do your network architecture diagrams clearly identify high-risk environments and data flows that may have legal compliance impacts?

213. Do you implement technical measures and apply defense-in-depth techniques (e.g., deep packet analysis, traffic throttling and black-holing) for detection and timely response to network-based attacks associated with anomalous ingress or egress traffic patterns (e.g., MAC spoofing and ARP poisoning attacks) and/or distributed denial-of-service (DDoS) attacks?

214. Do you publish a list of all APIs available in the service and indicate which are standard and which are customized?

215. Is unstructured customer data available on request in an industry-standard format (e.g., .doc, .xls, or .pdf)?

216. Do you provide policies and procedures (i.e. service level agreements) governing the use of APIs for interoperability between your service and third-party applications?

217. Do you provide policies and procedures (i.e. service level agreements) governing the migration of application data to and from your service?

218. Can data import, data export and service management be conducted over secure (e.g., non-clear text and authenticated), industry accepted standardized network protocols?

219. Do you provide consumers (tenants) with documentation detailing the relevant interoperability and portability network protocol standards that are involved?

220. Do you use an industry-recognized virtualization platform and standard virtualization formats (e.g.., OVF) to help ensure interoperability?

221. Do you have documented custom changes made to any hypervisor in use, and all solution-specific virtualization hooks available for customer review?

222. Do you provide anti-malware training specific to mobile devices as part of your information security awareness training?

223. Do you document and make available lists of approved application stores for mobile devices accessing or storing company data and/or company systems?

224. Do you have a policy enforcement capability (e.g., XACML) to ensure that only approved applications and those from approved application stores be loaded onto a mobile device?

225. Does your BYOD policy and training clearly state which applications and applications stores are approved for use on BYOD devices?

226. Do you have a documented mobile device policy in your employee training that clearly defines mobile devices and the accepted usage and requirements for mobile devices?

227. Do you have a documented list of pre-approved cloud based services that are allowed to be used for use and storage of company business data via a mobile device?

228. Do you have a documented application validation process for testing device, operating system and application compatibility issues?

229. Do you have a BYOD policy that defines the device(s) and eligibility requirements allowed for BYOD usage?

230. Do you maintain an inventory of all mobile devices storing and accessing company data which includes device status (os system and patch levels, lost or decommissioned, device assignee)?

231. Do you have a centralized mobile device management solution deployed to all mobile devices that are permitted to store, transmit, or process company data?

232. Does your mobile device policy require the use of encryption for either the entire device or for data identified as sensitive enforceable through technology controls for all mobile devices?

233. Does your mobile device policy prohibit the circumvention of built-in security controls on mobile devices (e.g., jailbreaking or rooting)?

234. Do you have detective and preventative controls on the device or via a centralized device management system which prohibit the circumvention of built-in security controls?

235. Does your BYOD policy clearly define the expectation of privacy, requirements for litigation, e-discovery and legal holds?

236. Do you have detective and preventative controls on the device or via a centralized device management system which prohibit the circumvention of built-in security controls?

237. Do you require and enforce via technical controls an automatic lockout screen for BYOD and company owned devices?

238. Do you manage all changes to mobile device operating systems, patch levels and applications via your company's change management processes?

239. Do you have password policies for enterprise issued mobile devices and/or BYOD mobile devices?

240. Are your password policies enforced through technical controls (i.e. MDM)?

241. Do your password policies prohibit the changing of authentication requirements (i.e. password/PIN length) via a mobile device?

242. Do you have a policy that requires BYOD users to perform backups of specified corporate data?

243. Do you have a policy that requires BYOD users to prohibit the usage of unapproved application stores?

244. Do you have a policy that requires BYOD users to use anti-malware software (where supported)?

245. Does your IT provide remote wipe or corporate data wipe for all company-accepted BYOD devices?

246. Does your IT provide remote wipe or corporate data wipe for all company-assigned mobile devices?

247. Do your mobile devices have the latest available security-related patches installed upon general release by the device manufacturer or carrier?

248. Do your mobile devices allow for remote validation to download the latest security patches by company IT personnel?

249. Does your BYOD policy clarify the systems and servers allowed for use or access on the BYOD-enabled device?

250. Does your BYOD policy specify the user roles that are allowed access via a BYOD-enabled device?

251. Do you maintain liaisons and points of contact with local authorities in accordance with contracts and appropriate regulations?

252. Do you have a documented security incident response plan?

253. Do you integrate customized tenant requirements into your security incident response plans?

254. Do you publish a roles and responsibilities document specifying what you vs. your tenants are responsible for during security incidents?

255. Have you tested your security incident response plans in the last year?

256. Does your security information and event management (SIEM) system merge data sources (app logs, firewall logs, IDS logs, physical access logs, etc.) for granular analysis and alerting?

257. Does your logging and monitoring framework allow isolation of an incident to specific tenants?

258. Does your incident response plan comply with industry standards for legally admissible chain-of-custody management processes and controls?

259. Does your incident response capability include the use of legally admissible forensic data collection and analysis techniques?

260. Are you capable of supporting litigation holds (freeze of data from a specific point in time) for a specific tenant without freezing other tenant data?

261. Do you enforce and attest to tenant data separation when producing data in response to legal subpoenas?

262. Do you monitor and quantify the types, volumes and impacts on all information security incidents?

263. Will you share statistical information for security incident data with your tenants upon request?

264. Do you inspect and account for data quality errors and associated risks, and work with your cloud supply-chain partners to correct them?

265. Do you design and implement controls to mitigate and contain data security risks through proper separation of duties, role-based access, and least-privileged access for all personnel within your supply chain?

266. Do you make security incident information available to all affected customers and providers periodically through electronic methods (e.g., portals)?

267. Do you collect capacity and use data for all relevant components of your cloud service offering?

268. Do you provide tenants with capacity planning and use reports?

269. Do you perform annual internal assessments of conformance and effectiveness of your policies, procedures, and supporting measures and metrics?

270. Do you select and monitor outsourced providers in compliance with laws in the country where the data is processed, stored and transmitted?

271. Do you select and monitor outsourced providers in compliance with laws in the country where the data originates?

272. Does legal counsel review all third-party agreements?

273. Do third-party agreements include provision for the security and protection of information and assets?

274. Do you provide the client with a list and copies of all subprocessing agreements and keep this updated?

275. Do you review the risk management and governanced processes of partners to account for risks inherited from other members of that partner's supply chain?

276. Are policies and procedures established, and supporting business processes and technical measures implemented, for maintaining complete, accurate and relevant agreements (e.g., SLAs) between providers and customers (tenants)?

277. Do you have the ability to measure and address non-conformance of provisions and/or terms across the entire supply chain (upstream/downstream)?

278. Can you manage service-level conflicts or inconsistencies resulting from disparate supplier relationships?

279. Do you review all agreements, policies and processes at least annually?

280. Do you assure reasonable information security across your information supply chain by performing an annual review?

281. Does your annual review include all partners/third-party providers upon which your information supply chain depends?

282. Do you permit tenants to perform independent vulnerability assessments?

283. Do you have external third party services conduct vulnerability scans and periodic penetration tests on your applications and networks?

284. Do you have anti-malware programs that support or connect to your cloud service offerings installed on all of your systems?

285. Do you ensure that security threat detection systems using signatures, lists or behavioral patterns are updated across all infrastructure components within industry accepted time frames?

286. Do you conduct network-layer vulnerability scans regularly as prescribed by industry best practices?

287. Do you conduct application-layer vulnerability scans regularly as prescribed by industry best practices?

288. Do you conduct local operating system-layer vulnerability scans regularly as prescribed by industry best practices?

289. Will you make the results of vulnerability scans available to tenants at their request?

290. Do you have a capability to rapidly patch vulnerabilities across all of your computing devices, applications and systems?

291. Will you provide your risk-based systems patching time frames to your tenants upon request?

292. Is mobile code authorized before its installation and use, and the code configuration checked, to ensure that the authorized mobile code operates according to a clearly defined security policy?

293. Is all unauthorized mobile code prevented from executing?

Appendix B

Cloud User Questionnaire

Here we show the CAIQ questions used in the cloud User questionnaire. These questions can be answered by Yes, No or Unknown choices (44).

1. Can the CSP accomodate with the tenant's data retention requirements?
2. Can the data be given to governments if requested for judicial requirements without informing the tenant or without constitutional guarantees?
3. Can the data be given to, shared with third parties, or used by the CSP for other purposes than the cloud service without the tenant's consent?
4. If the US-EU Safe Harbor applies, is the CSP registered?
5. Does the CSP use subcontrators?
6. Will the CSP inform the tenant of the subcontractors hired to provide the cloud service?
7. Will the CSP inform the tenant of any change in the course of the contract?
8. Does the CSP guarantee contractually to remain fully responsible for his engagements, even with the hiring of subcontractors?
9. Is the cloud service (including all its subcontractors audited by a third party?
10. If the cloud service is audited, are the scopes of the audits accurately defined?
11. Did the cloud service define an ISP (Information Security Policy) and obtain a security-related certification?

12. Is there a Tier certification of data centers (especially for physical availability and security) or equivalent certification?
13. Is the cloud service delivery managed under SLAs (Service Level Agreements)?
14. Does the CSP define and implement a business continuity plan?
15. Is the reversibility of the cloud service provided?
16. Does the CSP apply a segregation of duties in the CSP organization to protect the tenants?
17. If meta-data are extracted by the CSP from the process of tenant's data, are they used for the cloud service only?
18. Are the different authentication mechanisms to access the cloud service documented?
19. Are password policy enforcements well-defined and implemented?
20. Are secure password reset procedures well-defined and implemented?
21. Is the integration with the IAM of the tenant possible?
22. Is the integration with an ID-provider possible?
23. Are the identification and/or authentication of the devices used to access the cloud service possible as additional enforcement of the IAM?
24. Does the CSP document how the IAM of its employees related to the tenants' assets is performed?
25. Is data access of tenant user, tenant system administrator, CSP system administrator clearly defined?
26. Is data access of CSP employees, third party, other tenants denied?
27. Is IAM management and data access logging clearly defined and available?
28. Can the cloud service be provided as private or community?
29. In a multi-tenant system, are the data of the respective tenants segregated/isolated in such a way that it is technically impossible for any user of tenant A to receive entitlements to data of tenant B?

30. Are APIs developed in accordance with standards?
31. Are data integrity and security ensured for input and output?
32. Is the access to hypervisors management functions and administration consoles highly controlled?
33. Is data securely deleted from all storage media when the user's or tenant's account is deleted?
34. Does the CSP take defense-in-depth approach to wired or wireless network security?
35. Are sufficient controls in place at the hardware and virtual (if applicable) levels?
36. Are security mechanisms to prevent and analyze data leakage at the hardware and virtual (if applicable) levels available?
37. Are tools to prevent, detect and mitigate viruses and malwares at server stations available?
38. Is hardening process performed on the server stations?
39. Has the key management been defined through policies and procedures as required by the ISO/IEC27002:2013 standard?
40. Have the cryptographic mechanisms used for the cloud service been defined to guarantee adequate cryptographic strength?
41. Does the CSP use HSMs (Hardware Security Modules) for the protection of keys?
42. Is client-side encryption of data possible?
43. Is data-at-rest confidentiality ensured?
44. Is data-at-rest integrity ensured?
45. Can the backup retention plan be defined by the tenant?
46. Are backup controls defined and adequate?
47. Are tenants able to perform recovery tests, including reporting?

48. Does the CSP have a SIEM (Security Information and Event Management) for analyzing the security alerts and data logs?

49. Does the CSP have an adequate incident management procedure for managing and minimizing the impact of security incidents on tenants' data?

50. Does the CSP have adequate security policies and procedures regarding CSP employee security?

51. Is there a documented patch management process implemented in the cloud service?

52. Does the CSP test patches in acceptance environments prior to deployment?

References

[1] Nxt - The Blockchain Application Platform, author=NXT, howpublished = https://nxtplatform.org/, year=2017, note = Accessed: 2017-11-01.

[2] Pizza Hut suffers data breach; nearly 60,000 customers affected, author=CISOMAG, howpublished = https://www.cisomag.com/pizza-hut-suffers-data-breach-nearly-60000-customers-affected/, year=2017, note = Accessed: 2017-11-01.

[3] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 452–473. Springer, 2003.

[4] Abdelmageed Algamdi, Frans Coenen, and Alexei Lisitsa. Reputation system aggregation and ageing factor selection using subjective opinions classification. In *Global Summit on Computer & Information Technology (GSCIT17)*, 2017.

[5] Abdelmageed Algamdi, Frans Coenen, and Alexei Lisitsa. A trust evaluation method based on the distributed cloud trust protocol (ctp) and opinion sharing. In *International Conference on Computer Applications and Technology (ICCAT'17)*, 2017.

[6] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, **53**[4]:50–58, 2010.

[8] Marcella Atzori. Blockchain technology and decentralized governance: Is the state still necessary? 2015.

[9] Nuntapun Bhensook and Twittie Senivongse. An assessment of security requirements compliance of cloud providers. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 520–525. IEEE, 2012.

[10] Kirsimarja Blomqvist. The many faces of trust. *Scandinavian journal of management*, **13**[3]:271–286, 1997.

[11] Thomas Bocek and Burkhard Stiller. Smart contracts–blockchains in the wings. In *Digital Marketplaces Unleashed*, pages 169–184. Springer, 2018.

[12] George Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*. Dover Publications, 1854.

[13] Vitalik Buterin et al. Ethereum white paper, 2013.

[14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[15] VRBG Caldiera and H Dieter Rombach. The goal question metric approach. *Encyclopedia of software engineering*, **2**[1994]:528–532, 1994.

[16] Daniele Catteddu. Cloud computing: benefits, risks and recommendations for information security. In *Web application security*, pages 17–17. Springer, 2010.

[17] Davide Ceolin, Archana Nottamkandath, and Wan Fokkink. Subjective logic extensions for the semantic web. In *Proceedings of the 8th International Conference on Uncertainty Reasoning for the Semantic Web-Volume 900*, pages 27–38. CEUR-WS. org, 2012.

[18] CSA. Consensus assessments initiative questionnaire v3.0.1 (12-5-16 update)@ONLINE, 5 2016.

[19] Chris Dannen. *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*. Apress, Berkely, CA, USA, 1st edition, 2017.

[20] Chrysanthos Dellarocas. The design of reliable trust management systems for electronic trading communities. *URL:< citeseer. ist. psu. edu/380109. html*, 2001.

[21] Frank Doelitzscher, Christoph Reich, Martin Knahl, Alexander Passfall, and Nathan Clarke. An agent based business aware incident detection system for cloud environments. *Journal of Cloud Computing: Advances, Systems and Applications*, **1**[1]:9, 2012.

[22] Sheikh Mahbub Habib, Sebastian Ries, Max Mühlhäuser, and Prabhu Varikkattu. Towards a trust management system for cloud computing marketplaces: using caiq as a trust information source. *Security and Communication Networks*, **7**[11]:2185–2200, 2014.

[23] Einar Hille. *Analytic function theory*, **2**. American Mathematical Soc., 2005.

[24] Jingwei Huang and David Nicol. A formal-semantics-based calculus of trust. *IEEE Internet Computing*, **14**[5]:38–46, 2010.

[25] Jingwei Huang and David Nicol. A formal-semantics-based calculus of trust. *IEEE Internet Computing*, **14**[5]:38–46, 2010.

[26] Jingwei Huang and David M Nicol. Trust mechanisms for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, **2**[1]:9, 2013.

[27] Edwin Jacobs. Bitcoin: A bit too far? *Journal of Internet Banking and Commerce*, **16**[2]:1, 2011.

[28] Audun Jøsang. Artificial reasoning with subjective logic. In *Proceedings of the second Australian workshop on commonsense reasoning*, **48**, page 34. Citeseer, 1997.

[29] Audun Josang. Trust-based decision making for electronic transactions. In *Proceedings of the Fourth Nordic Workshop on Secure Computer Systems (NORDSEC'99)*, pages 496–502, 1999.

[30] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **9**[03]:279–311, 2001.

[31] Audun Jøsang. Subjective logic. *Book draft*, 2011.

[32] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, **43**[2]:618–644, 2007.

[33] Audun Jøsang, Xixi Luo, and Xiaowu Chen. Continuous ratings in discrete bayesian reputation systems. In *IFIP International Conference on Trust Management*, pages 151–166. Springer, 2008.

[34] Audun Jøsang, Stephen Marsh, and Simon Pope. Exploring different types of trust propagation. In *International Conference on Trust Management*, pages 179–192. Springer, 2006.

[35] Audun Jøsang and David McAnally. Multiplication and comultiplication of beliefs. *International Journal of Approximate Reasoning*, **38**[1]:19–51, 2005.

[36] Trevor I Kiviat. Beyond bitcoin: Issues in regulating blockchain tranactions. *Duke LJ*, **65**:569, 2015.

[37] Ron Knode and Doug Egan. Digital trust in the cloud: A precis for the cloudtrust protocol (v2. 0). *Computer Science Corporation*, 2010.

[38] Ronald Knode. Digital trust in the cloud, 8 2009.

[39] Ronald Knode and Douglas Egan. Digital trust in the cloud: Into the cloud with ctp – a precis for the cloudtrust protocol, v2.0. Technical report, CSC, 2010.

[40] Jin Li, Li-wei He, and Jian Liang. Distributed data storage using erasure resilient coding, November 1 2011. US Patent 8,051,362.

[41] Wenjuan Li and Lingdi Ping. Trust model to enhance security and interoperability of cloud environment. In *IEEE International Conference on Cloud Computing*, pages 69–79. Springer, 2009.

[42] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture. *NIST special publication*, **500**[2011]:292, 2011.

[43] Dejan Lukan. The top cloud computing threats and vulnerabilities in an enterprise environment, 2014.

[44] T. Martin. Modèle d'évaluation de sécurité cloud, 2016.

[45] Roger C Mayer, James H Davis, and F David Schoorman. An integrative model of organizational trust. *Academy of management review*, **20**[3]:709–734, 1995.

[46] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[47] Merriam-Webster. *Merriam-Webster's collegiate dictionary*. Merriam-Webster, 2004.

[48] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[49] Nils J Nilsson. Probabilistic logic. *Artificial intelligence*, **28**[1]:71–87, 1986.

[50] A Pannetrat. Ctp data model and api rev. 2.13. cloud security alliance, 2015.

[51] Cesare Pautasso, Erik Wilde, and Rosa Alarcon. *REST: advanced research topics and practical applications*. Springer, 2013.

[52] Marc Pilkington. Blockchain technology: principles and applications. *Browser Download This Paper*, 2015.

[53] Marc Pilkington. 11 blockchain technology: principles and applications. *Research handbook on digital transformations*, page 225, 2016.

[54] Teri Radichel. Case study: Critical controls that could have prevented target breach. *SANS Institute InfoSec Reading Room*, 2014.

[55] Archie Reed, Chris Rezek, and Paul Simmonds. Security guidance for critical areas of focus in cloud computing v3. 0. *Cloud Security Alliance*, pages 14–44, 2011.

[56] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial intelligence review*, **24**[1]:33–60, 2005.

[57] Glenn Shafer. *A mathematical theory of evidence*, **42**. Princeton university press, 1976.

[58] David Shrier, Weige Wu, and Alex Pentland. Blockchain & infrastructure (identity, data security). *MIT Connection Science*, pages 1–18, 2016.

[59] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J Alex Halderman. Security analysis of the estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 703–715. ACM, 2014.

[60] Aekkachai Sumetanupap and Twittie Senivongse. Enhancing service selection with a provider trustworthiness model. In *Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on*, pages 281–286. IEEE, 2011.

[61] Melanie Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[62] Aisha Thibodeaux. *Hacking back: Surviving in the digital age.* PhD thesis, Utica College, 2015.

[63] Søren Steffen Thomsen and Lars Ramkilde Knudsen. *Cryptographic hash functions.* PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Applied Mathematics and Computer ScienceInstitut for Matematik og Computer Science, 2005.

[64] Christoph Van der Elst and Anne Lafarre. Bringing the agm to the 21st century: Blockchain and smart contracting tech for shareholder involvement. 2017.

[65] Shou-Xin Wang, Li Zhang, Shuai Wang, and Xiang Qiu. A cloud-based trust model for evaluating quality of web services. *Journal of Computer Science and Technology*, **25**[6]:1130–1142, 2010.

[66] Andrew Whitby, Audun Jøsang, and Jadwiga Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proc. 7th Int. Workshop on Trust in Agent Societies*, **6**, pages 106–117, 2004.

[67] Aaron Wright and Primavera De Filippi. Decentralized blockchain technology and the rise of lex cryptographia. 2015.

[68] Linlin Wu, Rajkumar Buyya, et al. Service level agreement (sla) in utility computing systems. *IGI Global*, **15**, 2012.

[69] Hongwei Zhou, Wenchang Shi, Zhaohui Liang, and Bin Liang. Using new fusion operations to improve trust expressiveness of subjective logic. *Wuhan University Journal of Natural Sciences*, **16**[5]:376–382, 2011.

[70] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.