A Trust-based Elastic and Dynamic Authentication Solution for Multi-Cloud Computing

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy

in the Faculty of Science and Engineering

2023

By

Amina H. Gamlo

School of Engineering/Department of Computer Science

Supervisor: Dr. Ning Zhang

Co-Supervisor: Dr. Omaimah Bamasak

Advisor: Dr. Lucas Cordeiro

Blank Page

List of Contents

List of Figures

List of Tables

Abbreviations

CC	Cloud Computing
M2C	Multi-Cloud Computing
A3S	Authentication-as-a-service
DDEA	Dynamic Distributed Elastic Context-Aware Authentication
CSP	Cloud Service Providers
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
IaaS	Infrastructure-as-a-Service
AaaS	Authentication-as-a-Service
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service
MC	Multi-Cloud
IAM	Identity Access Management
MFA	Multi-Factor Authentication
VPC	Virtual Private Cloud
EBS	Amazon Elastic Block Store
RDS	Relational Database Service
EC2	Amazon Elastic Compute Cloud
DNS	Domain Name System
OTP	One-Time Password
PUK	Public Key
PRK	Private Key
SAK	Secret Access Key
AS	Authentication System
EDD	Elastic Dynamic and Distributed
NIST	National Institute of Standards and Technology
MAC	Message Authentication Code
CP	Credential Provider
RA	Registration Authority
CA	Certificate Authority
GC	Grid Computing
TPM	Trusted Platform Module
SP	Security Proxy
VoIP	Voice Over IP
R2	Message Authentication
R3	Continuous Authentication
CAM	Cloud Access Management
TCP	Trusted Computing Platform

UP	User Proxy
RP	Resource Proxy
AIK	Attestation Identity Key
TGT	Ticket-Granting Ticket
KDC	Key Distributed Center
HC	Host Cache
TRAVOS	Trust and Reputation model for Agent-based Virtual OrganizationS
IT	Interaction Trust
RT	Roles based Trust
WT	Witness Trust
CT	Certified Trust
LoA	Level of Assurance
END AuthN	Elastic and Dynamic Authentication
T-Cr	credential-based
T-Rp	reputation-based
T-Rc	recommendation-based
TruE	Trust Engine
LC	Local Confidence
GC	Global Confidence
LoA	Level of Assurance
AC	Authentication Context

Abstract

Continued advances in hardware, networking, and virtual machine technologies have led to the emergence of a variety of distributed computing schemes, such as Cloud Computing. The dynamic characteristics of cloud resources and services make them accessible to millions of users from anywhere on the Internet. Generally speaking, cloud services are specific to each cloud service provider (CSP) and unaware of other providers' resources servicing shared clients [2]. Multi-Cloud Computing (M2C) should allow users to benefit from various cloud-offered resources seamlessly. However, the open nature of CSPs and the independent planning and provisioning of resources for users make resource sharing in M2C challenging.

This research will focus on the authentication of M2C. Authentication is the foremost security measure in any computing system, as it validates the identities of communicating parties. Validating an identity usually requires proof of trust in a user or a system. Cloud Computing context presents many security challenges, especially regarding building trust due to the distribution of resources and multi-tenancy. Issues of standardization in network protocols and collaboration mechanisms add constraints and limitations to providing M2C authentication.

Current research provides a wide range of authentication solutions for M2C based on passwords, symmetric key cryptography, public key cryptography, and others. Each scheme presents an enhancement on others to provide secure, effective authentication for the distributed dynamic cloud services among multi-cloud services and users. This research examines a real-life use case scenario of M2C to describe a generic model. It also provides a threat analysis for M2C and specifies authentication requirements for M2C contexts. Based on the identification of requirements, an adaptive authentication solution is proposed that considers the special features of M2C. The proposed architecture is an Elastic and Dynamic Authentication (END) scheme. It is based on the dynamic (run-time) use of distributed resources available upon an authentication request from a claimant to build trust. Trust is not based on credentials only such as cryptographic keys and/or passwords; rather, it builds trust based on three modes of authentication: credential based, recommendation based, and reputation based. So, it offers on-demand authentication according to the level of trust required for the current context of the user.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification from this or any other university or other institute of learning.

Copyright Statement

Acknowledgement

I thank Allah for answering my prayers and guiding me through this journey.

I would also like to thank my wonderful supervisors: Dr. Omaimah Bamasak for her moral support, guidance and for being a wonderful friend whenever I needed one, and Dr. Ning Zhang for the continuous encouragement, brilliant scientific insights, and patience.

Special thanks to my partner in this journey, my husband, for his love and continuous encouragement and invaluable support throughout this long journey. After Allah's guidance, dearest Mustafa, I couldn't have done it without you.

My appreciation goes to my children, Aseel, Ahmad, Abdulaziz, Lama, and Muhammad for believing in me, making my life enjoyable, and keeping me on my toes always.

My dear sisters Ahlam, Eman, Tota, and Amjad, without your support and prayers, I couldn't have achieved my dream.

Finally, I offer my utmost gratitude to my parents, the guiding light in my life, for being my first teachers, for making me love science, for teaching me to never stop learning, and for always being there for me and keeping me in their prayers.

I would like to express my gratitude to my father (may his soul rest in peace). "I did it Dad". I wish you were here to celebrate this accomplishment with me. I hope I made you proud.

To all of you, a huge "Thank You" from the bottom of my heart.

Chapter 1: Introduction

1.1. Research Context

Cloud Computing (CC) is a well-established technology that continues to revolutionize the deployment of computing resources. Statista market research estimated that global cloud revenues would surpass $150 billion in revenue by the end of 2021 [73]. Cloud computing provides users with flexibility, scalability of infrastructure, reliability, sustainability, and cost-effectiveness [106]. On the other hand, CC presents many challenges, especially on the security front. Although the underlying concept of CC goes back to the 1950's, when mainframe computers offered shared computing resources to multiple users via terminal computers, the CC model demands more evolved measures to secure access to cloud services. CC requires special attention to security concerns like access control, authentication, confidentiality, availability, and integrity.

With the rapid explosion of cloud services offered by a growing number of Cloud Service Providers (CSP), Multiple Cloud Computing (M2C) or multi-cloud has gained increasing interest from industry and academia. Multi-cloud computing, a subtype of hybrid cloud, refers to users or organizations employing two or more cloud services from two or more vendors. According to Gartner, organizations want to avoid vendor lock-in and take advantage of the best IT solutions, leading to higher adaptation rates of M2C-based IT strategies [32]. The 2021 CC statistics show that enterprises continue to embrace multi-cloud and hybrid cloud strategies and are already using more than two public and two private clouds on average [59].

The increased growth of M2C adaptation stresses the fact that security across these diverse platforms is extremely important. Authentication is the first line of defense in any security provisioning solution. According to the NIST special publication 800-144 on Security and Privacy in Public Cloud Computing, most CSPs use the Security Assertion Markup Language (SAML) standard, which is an XML-based standard for exchanging authentication data between cooperating domains. Once a client is authenticated, a SAML transaction holds an identity token and information on the user's privileges, which in turn are verified by the service provider to grant the user the appropriate level of access [42], [106]. SAML request and response messages are typically mapped over Simple Object Access Protocol (SOAP), where the messages are digitally

signed using the private key associated with the public key certificate of the client. Although this method is currently accepted and used, it is vulnerable to certain attacks [42]. Secure Socket Layer (SSL) is widely used for secure internet communication including CC communication. Many studies show problems with non-browser SSL code leaving users open to Man-in-the-Middle-attack [42]. In addition, these methods allow users and organizations to be authenticated to receive multiple services limited by a single cloud operator [6]. In multi-cloud architecture, a user is required to sign multiple Service Level Agreements (SLAs), unique for each CSP, to receive services from multiple CSPs [4]. Hence, a customer is responsible for resource provisioning and brokering. For multi-cloud to work, authentication should be provided seamlessly between multiple services offered by multiple CSPs. Federated clouds make it easier by offering aggregated services offered by more than one CSP [4]. However, this limits the user's options to pre-set composite services offered by the cloud federation. Furthermore, these solutions are static in nature and unaware of all parameters present at run-time.

Current research and guidelines present many solutions, some of which consider some contextual attributes. This study focuses on understanding the features of Multiple Cloud Computing (M2C), with the intent to define a set of requirements for a comprehensive authentication solution. The requirements will be the basis of an elastic dynamic context aware authentication solution that takes into account M2C attributes and fulfils the formulated conditions.

1.2. Research Methodology

This research concentrates on devising an authentication solution for M2C, using context-aware computing, by applying the concepts of elasticity. By successful implementation of the attributes of this environment, the security of M2C will increase through many folds in the following manner [23]:

- A fine-grained authentication based on contextual attributes
- Deciding on the effectiveness of using elasticity to provide authentication

The main objectives of this research are as follows:

1. To describe real-life use case scenarios of M2C.
2. To construct a generic model for M2C service access.

3. To complete the threat analysis by identifying any threats related to identity and unauthorized access to data or services in the M2C environment.
4. Based on the outcome of the threat analysis, the aim is to specify the set of requirements, for a secure and efficient M2C authentication solution established on trust.
5. Reviewing the existing authentication solutions for distributed systems and to analyze them against the set of requirements, and further identifying the areas of improvement.
6. Designing and implementing a trust-based, reliable, secure, and efficient authenticated solution.
7. To evaluate the security of the designed solution.
8. To assess the performance of the accomplished outcome.
9. To publish the findings.

1.3. Research Question and Research Hypothesis

This research will address the following question:

RQ: How to facilitate effective authentication for multiple cloud computing (M2C) environments, with minimal performance cost?

M2C provides cost-effective, distributed, elastic, and variable resources to users. However, it also presents many security challenges due to its distributed, elastic multi-tenant nature. The M2C environment presents us with credentials and authentication methods already available and can be used to establish trust for the interacting parties at the time of authentication. Thus, the study proposes a trust-based dynamic elastic authentication solution. By "elastic authentication", we mean that a varying authentication assurance level is provided based on the provider's or user's requirements by supporting the use of a variable number and types of authentication factors when an authentication request is being made. An elastic authentication solution contemplates the special features of M2C and harnesses the power of changing parameters of M2C environment.

The hypothesis that is emphasized in this research will attempt to assess the following:

RH: By applying the idea of elastic authentication to M2C, it is possible to establish trust and provide authentication in a more effective manner with less performance costs.

Since M2C clients may roam from one network to another and connect to various entities, efforts to perform effective and efficient authentication may be reduced based on trust relations already established between cloud clients and those entities. The main idea of our solution relies on the dynamic use of authentication methods available when a user needs to be authenticated.

1.4. Novel Contributions and Publications

The main contributions of this work are listed as follows:

1. The description of a novel generic abstract model for multi-cloud solutions (**MC-Model**) is presented in Chapter 3.
2. Security Analysis of Multi Cloud Computing (M2C) based on the MC-Model.
3. Introduction of the concept of three modes of authentication (based on credentials, interaction history, and recommendations from other entities) are described in Chapter 5.
4. Chapter 5 presents the concept of Authentication-as-a-Service (A3S) which provides a distributed and dynamic on-demand authentication solution.
5. A novel framework for establishing trust in M2C entities (Trust Engine or TruE). It considers three trust establishing factors, including credentials provided by the claimant at the time of authentication, reputation of varied cloud entities (providers, services, and clients) based on their direct interactions, and recommendations of cloud entities based on collected entity evaluations of previous indirect interactions. This framework is introduced in Chapter 5.
6. Methodology for structuring LoA related factors in Multi Cloud Computing (M2C) is drafted in Chapter 6.
7. Algorithm for calculating aggregate LoA related factors based on credentials ($\boldsymbol{ALoA.Cr}$) for Multi Cloud Computing (M2C) entities for a given authentication instance is also presented in Chapter 6.
8. Chapter 6 also comprises of the three novel algorithms ($\mathrm{W.LoA_{Add}}, \mathrm{M.LoA_{Add}}, \mathrm{C.LoA_{Add}}$) for evaluating an aggregate value of LoA for independent credential based LoA related factors.

9. Further, Chapter 6 also presents a novel elastic, dynamic, and distributed authentication solution END AuthN for M2C based on trust establishment provided by (TruE) framework.

Publications:

The work presented in contributions 1, 2 was summarized in the following publication:

Gamlo, Amina H., Ning Zhang, and Omaimah Bamasag. "Mobile cloud computing: security analysis." 2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). IEEE, 2017.

1.5. Thesis Structure

The thesis is outlined as follows:

Chapter 1 provides the introduction in five sections, which include the research context, problem statement, methodology, novel contributions and publications, and an outline of the thesis.

Chapter 2 describes background information on Cloud Computing (CC) deployment and service models of CC, and Multi-Cloud Computing (M2C). It also presents the security issues and challenges of both CC and M2C contexts.

Chapter 3 classifies the authentication solutions in terms of authentication factor number and type. It also provides a literature review and in-depth analysis of authentication solutions for distributed systems in general and cloud computing in particular.

Chapter 4 depicts the how M2C works using a real-life use case scenario, along with its solution. These scenarios are the basis for a threat analysis, and the identification of general security requirements for M2C. It also introduces high level ideas for a novel authentication solution for M2C.

Chapter 5 reviews and analyzes existing models for establishing trust in computing environments comparable to M2C. It also provides the design of a novel framework for establishing trust, TruE, for interacting entities in M2C and the design and description of novel authentication solution for M2C based on trust.

Chapter 6 identifies credential based LoA-related factors and defines relationships between them. It also proposes a novel methodology for structuring these factors. It provides description and

evaluation of three novel alternate algorithms for combining these factors to compute aggregate value of LoA based on these factors under the additive rule. Chapter 6 also concludes the description of a novel authentication solution for M2C, END AuthN.

Chapter 7 presents an evaluation of the proposed END AuthN architecture, including evaluation methodology, evaluation against design requirements and against security risks, and a performance evaluation.

Finally, chapter 8 concludes the report and provides a view into future research plans.

Chapter 2: Background

2.1. Chapter Introduction

In order to provide a reliable authentication solution for multi-cloud users, some background information on the main concepts relating to the topic is described in this chapter.

This chapter introduces Cloud Computing (CC) and Multiple Cloud Computing (M2C). It provides an overview of CC, identifies its general characteristics, and describes its deployment and service models. It also introduces the security issues and challenges in this environment.

2.2. Cloud Computing: An Overview

Cloud Computing is no longer the future; it is the present. CC provides on-demand resources to users in a seamlessly distributed environment. Based on these concepts, the literature offers varying definitions for CC. Zeng et. al. lists the most expressive definition, stating CC as a network of parallel and distributed virtualized resources presented as a unit providing users with varying levels of IT service [104]. The National Institute of Standards and Technology (NIST) defines CC as, "a resource utilization mode that can allocate and deploy computer resources conveniently and immediately by network access according to the demand at the least management expense or with few activities executed by the supplier to carry out the resource release". CC is a pay-per-service on-demand mode of providing computational resources. The paradigm of CC is realized through virtualization. Virtualization refers to the ability of software to simulate an underlying infrastructure or hardware layer to keep the user working at a higher logical level and allow efficient utilization of the resources [97].

CC has many essential characteristics and features. Cloud service users commendably use many features like cost-effectiveness, variability of resources, seamless self-service, flexibility, reliability, location independence, and broad network access [106], [104], [61], [97]. CSPs also gain the advantages of scalability of infrastructure, cost-effectiveness, and sustainability [106]. One of the major benefits offered to both users and providers of CC is cost-effectiveness. For users, it permits the allocation of as many or as few resources as required to complete the tasks at hand. It also saves users the expense of allocating resources locally when they are underutilized. [97].

Furthermore, it reduces the cost of resource maintenance. It enables service providers to locate resources in low-cost real estate and/or in close proximity to low-cost power sources. [106]. In addition, the CC service paradigm is centered around better resource utilization, leading to long-term sustainability for the resource owner [106]. With a variety of delivery methods, CC enables the user to access various categories of services. Users can effortlessly request basic infrastructure services such as battery power preservation services, networking services, deployment platforms to run applications such as JAVA or Python or rented applications. Seamless self-service is yet another remarkable feature of CC. Users of cloud services can allocate the necessary resources automatically via simple user interfaces without the need for manual interaction with the CSPs. CC also offers users great flexibility and elasticity in service provision according to demand. Services can be scaled up or down by allowing users to acquire more, or to release redundant resources as they progress in executing their tasks. Users of cloud services value reliability, which is typically achieved by using many redundant service sites. This ensures business continuity and facilitates disaster recovery [106]. Another feature of CC is location independence. Users need not be aware of the exact location of resources, nor do they need to control them. Still, they can specify an abstract location, such as a country or city, if necessary [106]. Cloud services offer broad network access to users. Users also benefit from the standard mechanisms employed by the providers, allowing for diverse access platforms such as PDA's, laptops or mobile phones [97]. CSPs benefit from the scalability of infrastructure. It permits the network to expand or contract by adding or removing nodes and servers with minimal infrastructure and software modifications [106]. These key features make CC a popular means for providing computing resources.

2.3. Cloud Computing Deployment Models

There are different ways to deploy and manage service delivery to cloud users. These approaches (also known as deployment models) govern the allocation of resources and the relationships between cloud service providers and their customers. Four fundamental deployment models exist: public, private, community, and hybrid clouds. In the public cloud, infrastructure and computational resources are owned and managed by a CSP that offers Internet-based services to the public. The provider is presumably an external entity for all consumers. Private clouds, on the other hand, are owned and administered by the organization for exclusive resource provisioning. A third party may be assigned cloud management duties by the organization. The cloud may be

hosted either within the organization's datacenter or externally [42]. For private clouds, the organization has greater control over the infrastructure, users, and provided services. The community cloud is situated between the public and private clouds. In this model, the cloud infrastructure and services are tailored to a particular user community. Members of this community are affiliated with organizations that, among other things, share the same policy and security requirements [42]. The cloud infrastructure may be owned and operated by a third party or by one or more community organizations. The hybrid paradigm is a relatively more complex cloud deployment model. Hybrid clouds, as the name suggests, are comprised of multiple types of clouds that are perceived as distinct entities but are combined via a set of standards and rules that enable them to share data and applications [68]. Multi-cloud is a hybrid cloud variant in which computing resources are provided from multiple clouds without CSPs agreeing on how to share resources. With multi-cloud, the components are distinct cloud systems, as opposed to deployment strategies in hybrid clouds [37].

Different cloud deployment models have varying implications for the security and privacy of cloud users. Since private servers are owned or rented exclusively by the organization, security provisions remain internal. However, in the case of public clouds, security provision is managed by service providers, resulting in many challenging situations. As for community and hybrid clouds, security provision has the same circumstances as public clouds only for data and processes handled by public facilities.

2.4. Cloud Computing Service Models

While deployment models define resource allocation and cloud management, service models define service control and abstraction level. IaaS, PaaS, and SaaS are the three most prevalent service models [68]. IaaS refers to the provisioning of infrastructure capabilities such as storage, network bandwidth, processing capacity, and fundamental computing resources. Users obtain these resources as virtualized objects managed by a service interface, allowing them to choose the hosted operating system and development environment. [42]. The PaaS model enables users to develop and deploy applications on rented programming platforms, such as Python or JAVA, from the service provider. This model can help the user reduce costs associated with resource allocation and facilitate development and deployment processes. It will also enable users to manage the platform's applications and application environment settings [42]. The CSP will save the user the

expense and hassle of acquiring, storing, and managing the underlying hardware and software components, as well as the program and database development tools. In the SaaS model, services rendered are applications running on the cloud and accessed through a browser or other thin client [68]. This model reduces the total cost of hardware and software development, maintenance, and operations for users and the organizations they belong to [104]. Except for selecting utilization preferences and a few administrative settings, the user has no control over the underlying cloud infrastructure or individual applications [42].

As described in the previous paragraph, each service model gives the user a different level of control over cloud facilities, depending on the services rendered. According to NIST security and privacy guidelines, security provision depends on the amount of control given to CSPs and their clients (users or users' organizations) [42]. Figure 2-1 shows the five conceptual layers of any cloud: facility, hardware, virtualized infrastructure, platform, and application. Arrows on each side illustrate the amount of control on the user's side and provider's sides for each of the service models.

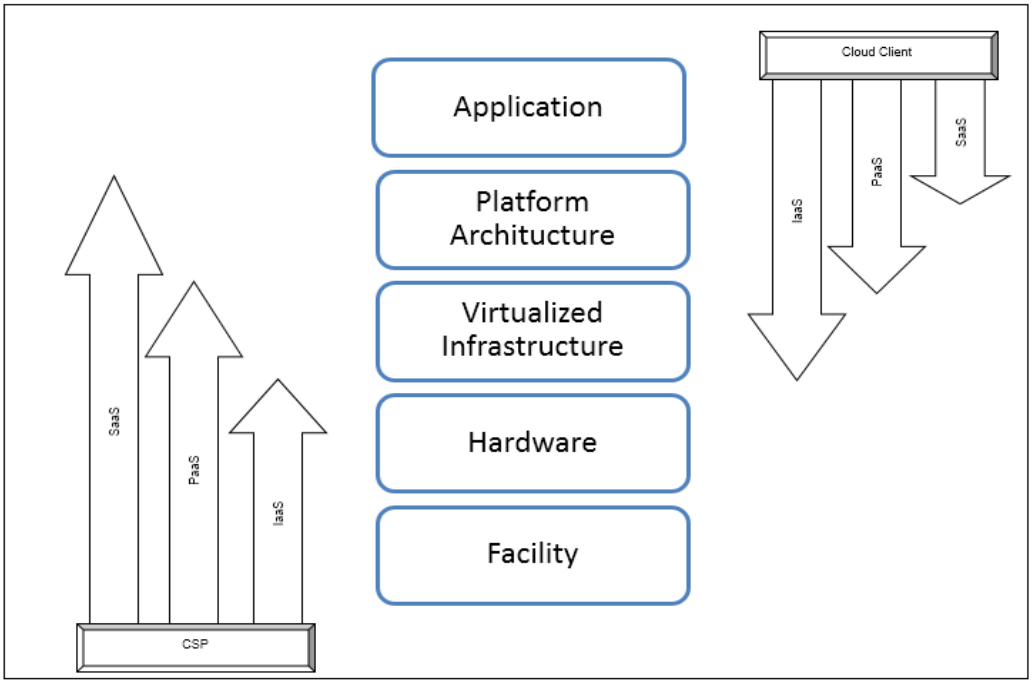

Figure 2-1. Difference in Control Between Client and Provider for Different Service Models

Figure 2-1 demonstrates that control at the client's side diminishes when more support is offered by the service provider [42]. Accordingly, under the security provision, the load is divided between CSPs and clients. Table 2-1 shows where the security burden falls for each service model.

Table 2-1: Security Provision Share between Clients and Providers for Different Service Models

Service Models	Responsibility of Security Provision by CSP	Responsibility of Security Provision by Cloud Client
IaaS	Only responsible for basic infrastructure	For all other layers
PaaS	For applications and application's environment	For all other layers
SaaS	Responsible for all layers	None

Although security provision for different cloud service models varies, it is apparent that the CC paradigm in general has a specific set of security challenges.

2.5. Multi-Cloud Computing (M2C)

Despite the significant growth in CC markets in recent years, there is a deficiency in standard programming interfaces [28]. This considerably limits the flexibility and portability of cloud services. Hence, a natural progression in CC is multi-cloud computing (M2C), where users or user organizations can utilize several cloud services simultaneously. M2C offers many key benefits [37]:

1. It helps users deal with peaks in service requests.
2. It offers solutions for cost optimization and better quality of service.
3. It allows users to take advantage of changes in services, offered by varying CSPs.
4. It helps users cope with added constraints, such as new locations or laws.
5. It improves the availability of resources and services.

6. It emancipates users from dependence on other CSP.
7. It allows for better disaster planning and recovery.
8. It allows users to choose the best services provided across different cloud markets.

2.6. Security Issues and Challenges

Along with the voluminous potential and promises of better resource utilization and client empowerment, CC presents many challenges. One study cites 66 research papers reviewed where security concerns were indicated as a major issue [37].

Security issues in the CC pose the biggest challenge for CSPs. According to Zissis and Lekkas, the context of CC security solutions should focus on two main issues: the placement of trust and the identification of security threats that are unique to CC [106]. In CC, the boundaries between an organization and outsiders blur, making it harder to identify trusted parties and locate security measures [106]. Identification of security threats is also necessary to implement any security system with the appropriate countermeasures.

A cloud platform's security system should guarantee confidentiality and privacy, service availability, data and application integrity, and recovery [42], [106]. Confidentiality ensures that information is not disclosed to unauthorized parties, whereas privacy ensures that individuals have control over who can collect or retain their information [83]. Availability ensures that authorized clients are not denied services [83]. Integrity affirms that the information and programs are only modified by authorized parties and in a specified manner [83]. While these requirements are not specific to CC security systems, the unique attributes of CC lead to specific risks that should be addressed [106]. This is due to delegating management of data and/or processes from the owner or user to the CSP [55]. Also, since this deputization is typically communicated over different types of networks, protection is necessary for the communication path between users and providers [106]. Further, the cloud environment is usually implemented in a distributed manner, posing more challenges for the security system. Furthermore, the multi-tenancy nature of CC leads to an increased number of access points to the system, leading to accrued confidentiality and privacy risks. Another issue is object reusability, which may lead to an intentional or unintentional confidentiality breaches due to data reminiscence [106]. Data reminiscence refers to the residual representation of data that has been removed. As for integrity, there is an increased risk of insider

attacks on asset's integrity due to the increased number of entities involved [106]. Moreover, the availability of varying services offered by the service providers is the heart of CC. Extra measures need to be taken to ensure continuous on-demand service provision despite the possibility of misbehaving users or security breaches occurring. All these attributes require a reliable security system that considers these issues.

A major goal of a security system at CC is protection against threats. Security threats vary depending on the level of service provided and the layer of the cloud at which the user operates. Table 2-2. enumerates general security requirements at each service level and possible threats [106].

Table 2-2: Security Requirements and Threats at each Service Level [106]

<table>
<tr><th>Service Levels</th><th>Possible Threats</th><th>Security Requirements</th></tr>
<tr><td>SaaS</td><td>• Data Interception
• Modification or deletion of data
• Privacy breach
• Impersonation
• Session hijacking
• Traffic flow analysis</td><td>• Privacy
• Data protection
• Application protection
• Access control
• Service availability
• Communication security</td></tr>
<tr><td>PaaS</td><td rowspan="2">• Software modification or deletion
• Impersonation
• Session hijacking
• Traffic flow analysis
• Connection flooding
• Denial of Service
• Disrupting communication
• Defacement</td><td rowspan="2">• Application protection
• Data protection (in transit, stored, reminisce)
• Virtual Cloud protection
• Cloud management control
• Communication security</td></tr>
<tr><td>IaaS</td></tr>
<tr><td>Physical level</td><td>• Connection flooding
• Denial of Service</td><td>• Hardware security
• Hardware reliability
• Network Protection</td></tr>
</table>

	• Network Attacks • Hardware theft or modification • Natural disasters	• Network Resources Protection

Although users do not operate at the physical level, threats occurring at this point may affect service provision at other levels. As the above table demonstrates, that the security objectives of CC should include the characteristics of a distributed system, complimented with the cloud's special features, as discussed. Accordingly, the security objectives of CC should include:

- Authentication of all communicating parties.
- Availability of data in motion or stored.
- Integrity of data in transit or stationery.
- Integrity of applications to ensure correct operation.
- Access control to all assets and services, or parts of them.
- Confidentiality of data held by cooperating systems.
- Maintaining physical level security when adding or removing resources.
- Clear separation between data and processes at the virtual level.

It should be noted that the first objective of having a dependable authentication solution will play a significant role in meeting the other subsequent tasks. A reliable authentication system will lead to well-guided access control of the system's resources. Also, data and processes will only be disclosed to authenticated parties, ensuring the confidentiality of the data and processes. Moreover, it will also guard against misuse or modification of data and applications by unauthenticated parties. Mostly all organizations have an authentication framework for non-cloud operation, but it's not necessary that these frameworks extend their cloud services easily [104]. Alternatively, an organization may need to utilize two different authentication systems, one for internal organizational systems and the other for cloud-based systems, possibly leading to further complications [104]. For M2C, the aim is to allow the utilization of multiple clouds instead of complete dependency on a single cloud, leading to distribution of reliability, trust, and security among CSPs [19]. However, security provision becomes more complicated with interacting services and entities for identification and trust [19], [5].

Hence, a reliable authentication solution is essential to minimize most security threats, allow sharing of digital identities and attributes across both cloud and non-cloud domains, and to provide a clear separation of the entities managed in M2C environment.

2.7. Chapter Summary

This chapter has presented background information concerning CC, along with its concepts and features. It also delivered an overview of CC deployment models and service models, pointing out implications for the provision of security. Later, M2C along with its definition and benefits, is discussed. Finally, the chapter reviews the security challenges presented by CC in general and MC specifically.

The next chapter reviews existing authentication solutions for computing environments comparable to MC in order to identify research gaps and create a road map for the proposed solution in our work.

Chapter 3: Authentication Solutions: A Literature Survey

3.1. Introduction

Authentication is the primary line of defence and a building block in any security solution. User authentication provides a basis for reliable access control, confidentiality, and integrity of a system.

In this chapter, we provide background information on authentication (Section 3.2). The remaining part of the chapter will deliver a classification of authentication solutions. Section 3.3 categorizes authentication solutions according to the type and the number of factors used to identify an applicant. Section 3.4 provides a brief look at context-aware authentication solutions. A literature review of authentication solutions for Distributed Systems including Peer-to-Peer systems, Grid Computing, and Cloud Computing is presented in Section 3.5.

3.2. Authentication Basics

Authentication is one of the main security objectives for any information system. It is one of the main five categories of security services outlined in X.800 [83]. This section provides definitions for authentication types, highlights their importance to any security system and explains the generic process of authentication, and introduces authentication tokens that may be used. We also identify threats to the authentication process.

3.2.1. An Overview

X.800 recommendation outlines five security services, the first of which is authentication. There are two types of authentication: peer entity authentication and data origin authentication [83]. Peer entity authentication provides validation of the identity of a peer entity connecting with another via the same protocol. It is usually provided at the establishment of a connection, or during the data transfer phase. Data origin authentication, on the other hand, provides validation of the source of a data unit, with no guarantees against duplication or modification of data units. Various

Cryptographic techniques are usually employed to provide authentication for a system, such as Digital Signatures, Hash algorithms, and Message Authentication Code (MAC).

Authentication is the first line of defence in any security system. It provides evidence to guide the decision to allow access to any of the system's assets. This decision will provide secrecy, or confidentiality, of data and processes against any illegal parties, which are eliminated by the authentication process. It will also safeguard the data and processes against modification by unauthenticated parties. Authentication also shields against malicious drainage of system resources, which ensures the availability of system assets to rightful users.

3.2.2. Authentication Process

The process of authenticating an applicant to any system consists of two phases: first, is the identification, and then actual authentication [89]. Identification is the step where the applicant provides proof, in the form of a token, of his/her claimed identity to the system. Authentication is the step when the system validates the applicant's identity by referring to an authentication authority, maybe a database of all identities to check the evidence, or a token provided in the first step. Figure 3-1 shows the steps of the authentication process. This typically is implemented through an authentication protocol message exchange, usually referred to as an authentication protocol run. The authentication protocol run results in either successful authentication of the applicant or authentication failure.

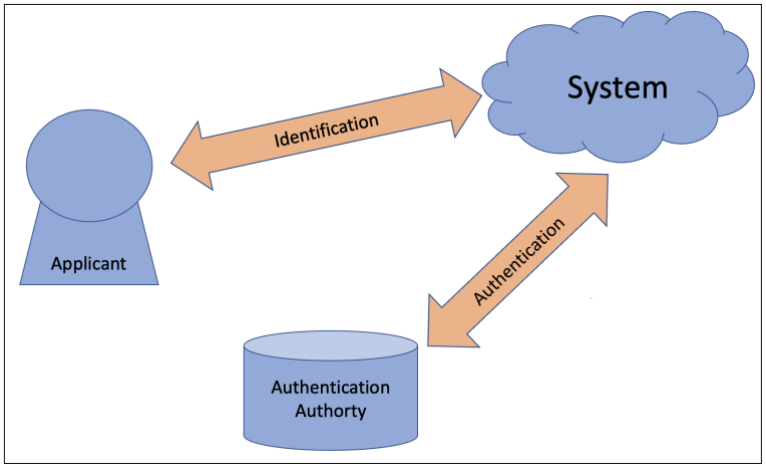

Figure 3-1. The Process of Authentication

According to NIST's electronic authentication guidelines, these tokens or credentials are issued by a Credential Provider (CP) trusted to issue and register electronic tokens and credentials to

subscribers [89]. A CP may be an independent third party, or it may issue credentials for its own use. An applicant needs first to apply to become a subscriber of a CP, which is done through a Registration Authority (RA) trusted to establish and vouch for the identity or attributes of an applicant to a CP [16]. The RA may be an integral part of the CP, or it may be independent. A verifier, on the other hand, is an entity that verifies the identity of the applicant by confirming his possession and control of a token via an authentication protocol. To accomplish this, the verifier may also need to validate and verify the status of the credentials that link the token and identity [16].

To ensure that the authentication process is reliable, the following steps need to be carried out [16]:

- Risk assessment needs to be employed for the system assets.
- Based on risk assessment, a security level for the intended system should be defined for all system assets. The risks are mapped to a specific assurance level.
- A suitable technology (or technological tools) should be matched to the required assurance level.
- The implemented security system, then, needs to be validated against the required assurance level.
- Periodical reassessment is necessary to update requirements and employ more advanced technology.

This process allows users and organizations to be confident with respect to the reliability of the systems they interact with. However, it is a static preset approach that is unable to cope with dynamic computing contexts of operations such as Cloud Computing or Mobile Computing. In these contexts, risk levels and operational parameters change at run-time requiring systems to adapt and handle the new risks and take advantage of context parameters at any given time.

3.2.3. Registration and Identity Proofing

Registration and identity proofing are the first areas of concern for defining security requirements. Going through this process, the user is referred to as an applicant, a subscriber, or a claimant depending on the stage of the process. Registration usually precedes the authentication process illustrated in Section 3.2.2. It is a process by which an applicant applies to become a Subscriber of a CP to receive identity credentials. Whenever a user uses these credentials to be authenticated, he is referred to as a claimant. Figure 3-2 illustrates the user's transition through this process.

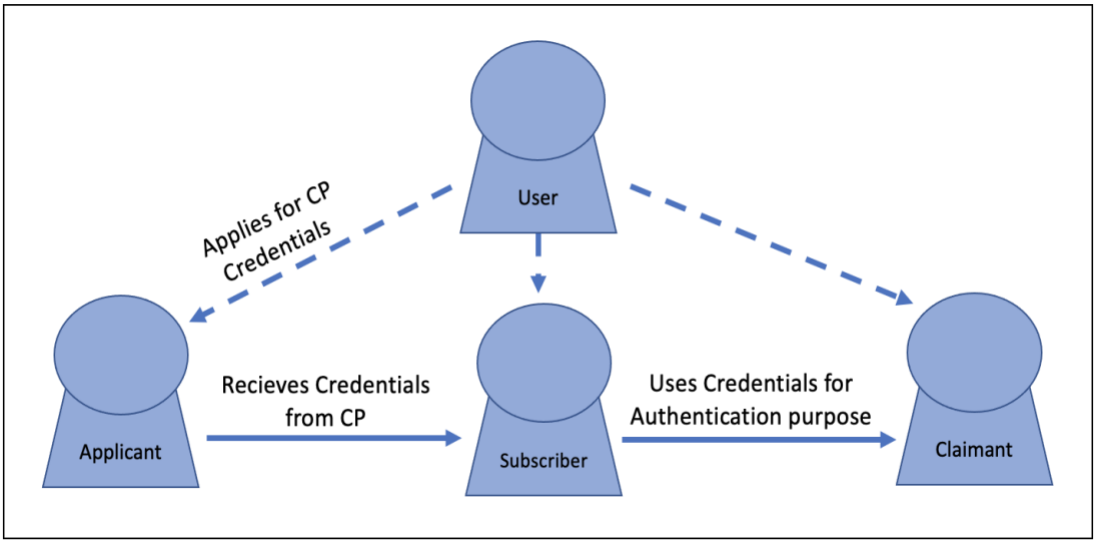

Figure 3-2. User Role during the Authentication Process

Identity Proofing is a process by which a CP and an RA collect and verify information about a person for the purpose of issuing respective credentials to that person. Figure 3-3 describes the registration process in general.

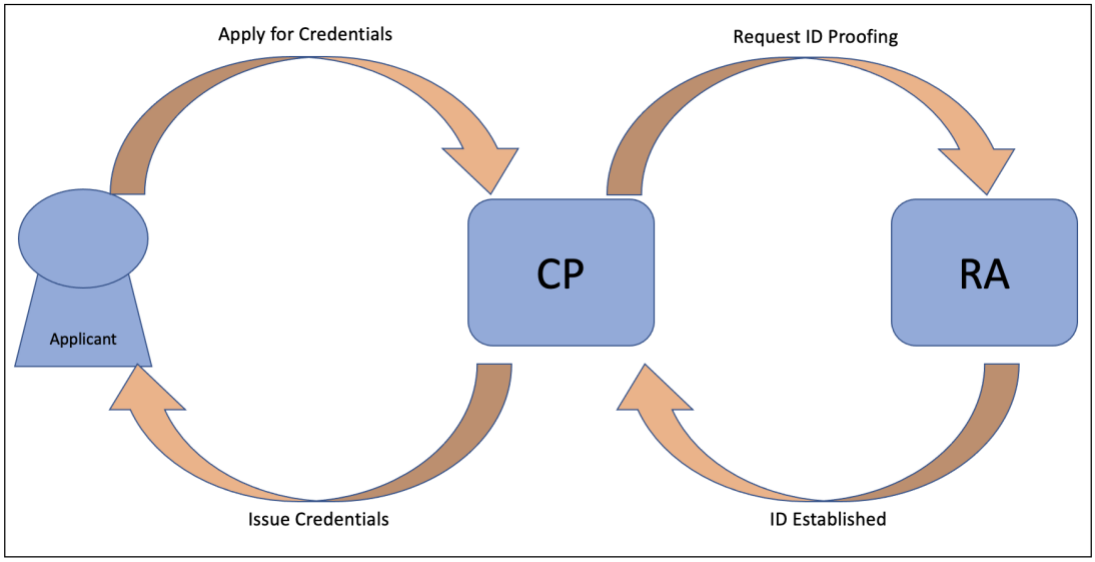

Figure 3-3. Registration and Identity Proofing Process

The registration and identity proofing process should ensure [16]:

- Existence of an entity with the applicant's attributes that are sufficient to uniquely identify this entity.
- The entity entitled to the identity is, in fact, the Applicant whose token was registered.
- Using this token in the future, it is difficult for the claimant to repudiate the registration and dispute authentication.

3.2.4. Authentication Tokens

A token is defined as something the claimant has and controls, (typically a cryptographic module or password) that is used to authenticate his/her identity. It typically contains a secret value, or token secret, used to generate authenticator outputs on demand to verify the claimant's possession of the token. Tokens are based on one or more authentication factors [16]:

- Something you know (knowledge tokens): such as a password.
- Something you have (ownership tokens): such as an ID card.
- Something you are (identity tokens): such as a person's fingerprint.

Hence, a token can be hardware, software, or some information you remember. Tokens are also characterized by the number of factors they use. Accordingly, there are two types of tokens:

- Single-factor token, which depends on one authentication factor, such as a password.

- Multi-factor token, which employs two or more authentication factors, such as a smart card and a Personal Identification Number (PIN) to activate it.

Tokens can be further classified as:

- **Memorized Secret Token**: It is a token of knowledge. It is a secret shared between the subscriber and the CP, typically a character string (password) or a numerical string (PIN). For this type, the token authenticator is the secret itself.
- **Pre-registered Knowledge Token**: This is another type of knowledge token which is comprised of a series of responses to a set of prompts. The series of responses is the shared secret, which is typically established by the subscriber and CP during the registration process. The prompts may be pre-registered questions or images. The token authenticator in this case is the set of memorized responses (answers to questions or choice of images).
- **Look-up Secret Token**: It is an ownership token that stores a set of secrets shared between the claimant and the CP. The verifier provides the claimant with an input value for the token. The input is used by the claimant to look up the correct secret to respond to a prompt. An example of this is a card token, that stores input strings and corresponding outputs when a verifier prompts the claimant with an input value, the claimant inputs the string into the token. A look-up operation results in the corresponding string of characters on the token [16]. The token authenticator for this type is the output of the look-up process.
- **Out of Band Token**: It is also an ownership token. It is a physical token that is uniquely addressable and can receive a secret selected by the verifier for one-time use. The device is possessed and controlled by the claimant and can communicate privately over a channel separate from the primary channel for authentication. The process usually starts with a claimant attempting to access a website or a resource. He, then, receives a text message on his preregistered phone with a one-time authenticator to be presented to the verifier using the primary channel for authentication. The token authenticator is the received secret.
- **Single-factor (SF) One-Time Password (OTP) Device**: It is another type of ownership token, which is a hardware device that has an embedded secret to be used as the seed for generating one-time passwords. The token authenticator is the one-time password generated by the device.
- **Single-factor (SF) Cryptographic Device**: It is an ownership token, which is a hardware device that uses embedded symmetric or asymmetric cryptographic keys to perform cryptographic

operations on input provided. Authentication is achieved by demonstrating ownership of the device. The token authenticator, which is usually a signed message, is dependent upon the cryptographic device and protocol.

- **Multi-factor (MF) Software Cryptographic Token**: It is a multi-factor token of an ownership factor that is activated by a knowledge factor or an identity factor. It is in the form of a cryptographic key stored on disk or "soft" media. The key can only be activated using a second factor of authentication, such as a fingerprint or a password. The token authenticator is dependent on the cryptographic protocol, which is usually a signed, message.
- **Multi-factor (MF) OTP Device**: It is also a multi-factor token of an ownership factor that is activated by a knowledge factor or an identity factor. It is a hardware device that generates one-time passwords upon activation through a second factor of authentication, such as a fingerprint or a password. The one-time password is either displayed to the claimant and then manually entered as a password by the verifier, or it is entered from the device to a computer directly. The token authenticator in this case is the one-time password.
- **Multi-factor (MF) Cryptographic Device:** It is a multi-factor token of an ownership factor that is activated by a knowledge factor or an identity factor. It is a hardware device containing a protected cryptographic key that requires activation using a second authentication factor, such as a PIN or a fingerprint. The token authenticator depends on the specific cryptographic device and protocol and is usually a signed message.

3.2.5. Threats on the Authentication Process

The goal of the authentication process is to establish the identity of the claimant through an authentication protocol message exchange, during or after which a protected session is established for further data exchange. Several threats pose themselves against the authentication process, which requires management mechanisms at both ends to secure the authentication activities. These threats are [16], [89]:

- On-line guessing, where an attacker attempts to guess the value of the token authenticator through repeated login trials.
- Phishing is when a subscriber is tricked into divulging his/her token secret, sensitive data, or authenticator values, which can be used to impersonate the subscriber later. This occurs by luring

the subscriber to interact with a fake verifier through a fraudulent email redirecting him/her to a fake verifier.

- Pharming is where a subscriber is re-directed to a fraudulent website through manipulation of the domain name service or routing tables during his/her effort to connect with a legitimate verifier.
- Eavesdropping is where an attacker listens passively to the authentication protocol to capture login information to be used in a subsequent active attack.
- Replay is where an attacker can capture authentication messages between a legitimate claimant and a verifier, and then replay them at a later time to authenticate as that claimant.
- Session hijack is where the attacker places himself /herself between a subscriber and a verifier at the beginning of the authentication protocol message exchange, allowing him/her to pose as a subscriber to the verifier/RP or as a verifier to the subscriber to control session data exchange.
- Man-in-the-middle is a threat where an attacker inserts himself between the claimant and verifier to gain access to the authentication protocol messages. Then the attacker can impersonate the verifier for the claimant while simultaneously impersonating the claimant for the verifier. This may allow him/her to authenticate himself/herself to both parties successfully.
- Denial of Service attack is during which the attacker overwhelms the verifier with authentication requests.
- Malicious code attack, which exploits authentication tokens.

3.3. Classification of Authentication Solutions based on Number of Factors

Authenticating a user of a system is accomplished via an authentication process, as explained in Section 3.2. The goal of user authentication is to verify the user's claimed identity. This verification is based on one or more authentication factors. The authentication factor(s) are materialized in a token of various types, as discussed in Section 3.2.4. Thus, there are four general methods to authenticate an identity [83]:

1. Something the user knows where authentication is based on the user providing proof that he/she knows a shared secret, such as a password or a PIN.
2. Something the user owns, where authentication is achieved upon the user providing proof that he/she possesses a token of a shared secret such as a cryptographic key or a smart card.
3. Something the user is, which may be referred to as static biometrics, such as recognition by fingerprint or retina.

4. Something the individual does, which may be referred to as dynamic biometrics, such as recognition by voice pattern or handwriting characteristics.

Using one of the above methods is considered a single-factor authentication. Combining two or more authentication methods is referred to as multi-factor authentication. The following sections will provide a literature review of authentication solutions for each type.

3.3.1. Single-factor Authentication Methods

As mentioned earlier, single factor authentication is based on validating the identity of a user by employing one authentication factor of any type: knowledge, ownership, static identity, or dynamic identity. The following sections will present authentication solutions in recent research for each type.

3.3.1.1. Knowledge Factors

For this type of authentication, the user needs to present evidence that he knows some shared secret. In this case, the shared secret represents the authenticator value of the authentication token. There are two tyes of tokens that use knowledge authentication factors (section 3.2.4): memorized-secret tokens and pre-registered knowledge tokens. Passwords and PINs are examples of Memorized-secret tokens. The use of pre-registered questions or images is an example of pre-registered knowledge token.

Passwords are the most used authentication token. However, passwords are not very secure since they are vulnerable to several types of threats [16]:

- Passwords can be copied from soft (disc) or hard (paper) media.
- Passwords can be learned by watching a keyboard entry or employing keystroke logging software (eavesdropping).
- Passwords may be revealed to a fake website in a pharming attack.
- Passwords may be disclosed to a fraudulent website through the attacker's email (phishing).
- An Online Dictionary may be used to guess the password.
- The password may be revealed to an officemate or someone impersonating a system administrator.

Thus, plenty of authentication solutions propose the use of passwords with improvements to overcome the above-mentioned weakness. One study claims that password-secure systems are usually unfair to users and unable to identify the source of the problem when there is a breach [20]. They propose an elaborate six-stage authentication scheme where the user and the system use prime numbers to generate dynamic passwords based on public parameters chosen at the initial stage. This system is fair to users as it allows them to choose passwords and find out the cause of a system breach without unfair consequences to the user. It claims to be a safe system as it guards against the disclosure of passwords to administrators [20]. However, it doesn't consider network security and key exchange policies. Ren & Wu propose a dynamic password authentication based on OTP that considers time and space [72]. Downloadable free software can generate the password based on the user's static password, time factor as well as the computer's physical address MAC. This scheme has low overhead and added security to counter a Man-in-the-Middle-Attack [72]. It is still susceptible to phishing attacks on password generation software. Password generation overhead may be unsuitable for users with mobile devices.

As for pre-registered knowledge tokens, they can be discovered through social engineering where responses are learned by the attacker through social media or acquaintances of the user. Moreover, they are subjected to the same risks that threaten passwords. However, they present a bigger challenge for an attacker since he needs to gain knowledge of several responses as opposed to one password.

3.3.1.2. Ownership Factors

For this method of authentication, the user must prove that he owns a token of authentication. Proof of ownership is based on providing the secret output of the token. Look-up secret tokens, out of band tokens, Single-factor (SF) OTP devices, and SF cryptographic devices are examples of ownership factors. Public key (PK) authentication is also classified as an ownership factor since it requires proof of ownership of the PK certificate. All these tokens are subject to the following threats [16]:

- Theft of the look-up token, device, or mobile phone used for out of band authentication.
- Copying the authenticator output value.

PK authentication is very commonly employed for many computing applications as it is well established and able to utilize many cryptographic algorithms that have proven to be strong. One PK solution is discussed for Voice over IP (VoIP) applications with three stages: registration, certification, and periodic authentication of the client [48]. In this solution, the proxy server and the registrar server handle client authentication through PK certificates and a One-Time key shared between the proxy server and the client. Two disadvantages of this approach are hashed non-memorizable passwords, and the inefficient requirement of periodic authentication, where the proxy server needs to establish a secure channel with every client in session [48].

3.3.1.3. Static Identity Factors

For the static identity factor, the user must provide proof of his biometric token. He needs to be positively identified through a scanned fingerprint, or retina print, or any sort of static biometric. In all the above cases biometrics are usually collected during the registration and identity proofing processes before authentication. In many scenarios, biometrics are collected in-person as they require special hardware equipment, and are intrusive to the user [26]. Biometrics is under threat of being replicated from a copy. Although it is a more challenging attack, it still poses a risk [72]. Furthermore, compromised biometrics are irreplaceable [84].

Biometric tokens are not commonly used due to their intrusiveness to users and high expenses [72]. One solution proposes a framework of user authentication based on fingerprint scans stored on smart cards [99]. They chose fingerprints due to the availability of the matching algorithms and the advancement in fingerprint technology. Another biometric authentication solution recommends the use of the BioCapsule (BC) concept [84]. BC is generated based on the difference between the captured biometrics of the user and those of a Reference Subject (RS), where the user's biometrics are captured and fused with the RS biometrics to extract secure keys [84]. The solution has many merits with limited application.

3.3.1.4. Dynamic Identity Factors

For the dynamic identity factor, the user should provide proof of his dynamic biometric token, which is "something he does" rather than static biometrics of "something he is". He needs to be positively identified by his voice, handwriting, or another behavioral characteristic. Like static biometrics, dynamic biometrics can also be replicated from stored copies.

Dynamic identity factors are also intrusive to users and may be expensive. Hence, they are not commonly used. One dynamic ID authentication scheme suggests a pattern-matching voice recognition scheme [50]. They suggest this scheme for enhancing Internet service security. It relies on a Secure Voice Biometric Server to generate, train, and update the user's voiceprints, store them, and perform the matching algorithm [50]. The advantages of the proposed system are ease of upgradability, heavy matching computations confined to the server, and variability of security levels. Another voice-based login scheme is suggested for Linux OS which overcomes the threat of record and replay [82]. It is accomplished by a speech-to-text conversion with 80%-85% accuracy [50]. The 'Record and Replay' threat is counteracted using random passphrases [82]. Another piece of research is based on developing a system of biometrical identification based on handwriting dynamics [58]. It generates cryptographic keys that contain dynamic characteristics of handwritten passwords using software products for recognizing handwritten signatures. Another biometric of interest is keystrokes dynamics on mobile devices. Trojahn and Ortmeier suggest a mobile authentication system based on keystroke dynamics [91]. They list keystroke features that can be used to augment a password authentication scheme. Learning process of the user and variability due to injury or time are clear limitations of this system [91].

3.3.2. Multiple-Factor Authentication Methods

Combining two or more factors in an authentication scheme is a threat mitigation strategy [16]. When an attacker needs to guess a password and steal a token, the effort to breach the system is much harder. For this reason, several research studies suggest multifactor authentication in various contexts. Kim and Hong introduced multi-factor authentication that combines tooth imaging and voice recognition for mobile devices [49]. The authentication process relies on a weighted summation operation of the combined metrics with a simple structure and excellent performance [49]. More examples of Multiple factor authentication are presented in Section 3.5.

3.4. Context-Aware Authentication Methods

Context is defined as "any information that can be used to characterize the situation of an entity" [74]. The goal of context-aware computing is to obtain and use information in the context of any application and then provide services accordingly. As for authentication in different computing contexts, there are many research efforts that provide context-aware methods. For Long Term

Evolution (LTE) Mobile networks, Purkhiabani and Salahi propose an authentication method that considers the traffic and storage limitations of the mobile context to improve the performance of the Authentication and Key Agreement (AKA) protocol [70]. This method was based on sharing key vectors between the user's Mobile Station (MS) and the Home Subscriber Server (HSS), which leads to less information transfer and less storage. In the same context, another method is suggested to improve AKA for mobile devices in 3GPP networks [33]. This method avoids the double execution of AKA at the network and service layers by binding the two with the IP Multimedia Private-user ID (IMPI) number to improve energy consumption on energy-sensitive mobile devices [11, 33]. It will also provide a better defence against DOS attacks.

3.5. Distributed Systems Authentication Methods

M2C is a paradigm for providing computing resources where interacting entities, including users and Cloud resources are distributed. Hence, this section will examine authentication methods for other types of distributed systems, such as Peer-to-Peer systems (Section 3.5.1) and Grid Computing (Section 3.5.2). Further, Section 3.5.3 will examine and provide an analysis of authentication methods for M2C. Thus, it will assist in evaluating the current research against M2C requirements, so that a gap is identified to decide on the best way forward for this research.

3.5.1. Peer-to-Peer Authentication Methods

P2P systems are characterized by partitioning workloads between equally privileged node peers participating in an application. A few authentication methods for P2P context are summarized in the following points:

- Encrypted PWD (Enc PWD) [11]: Skype is a P2P application, where each Skype client (SC) listens on certain ports for incoming calls and maintains a host cache (HC), which is a table of super Skype nodes IPs and ports and buddy nodes. Skype has a central login server that stores usernames and passwords. Login communication is encrypted using AES symmetric key cryptography to secure confidentiality. AES key exchange is secured through RSA public key cryptography. Upon login, HC and buddy lists are populated through the login server. To search for another user, SC sends TCP packets to Super nodes or other clients on the buddy list [11, 44, 54]. The search is cascaded through other clients. Authentication occurs at login, where SC enters

the username and password. Although research states that all Skype communication is encrypted, the study doesn't comment on node-to-node authentication during the population of the buddy list.

- PKI based [44]: Authentication is accomplished by confirming the authenticity of a PUK certificate through a group of distributed TTPs acting as a Certificate Authority (CA). This method takes advantage of the decentralized nature of P2P networks as authentication functionality is relocated to clients and the new authentication servers. This method is based on three protocols: App server-setup, client authentication, and client access to the app server. In the App server-setup, the app server enlists the authentication servers sharing parts of PRK_i, where the corresponding PUK_i is sent to the app server. In the client authentication protocol, the client receives an authentication token that is certified by PRK_i, and all authentication servers partially sign the token. It is based on the idea of establishing trust based on a certain threshold, in this case, the number of authentication servers involved. The client access protocol is used to authenticate the client to the app server using the authentication token. The partitioning of authentication functionality between clients and authentication servers allows for the distribution of trust and supports scalability. However, it incurs more weight on clients in the authentication process.

- IBC [54] [65]: or ID based Cryptography scheme utilizes any identifying string of the user, such as an email or an IP address, as the public key. In this scheme, a PRK generator (PKG) is needed to generate PRKs from the designated PUKs leading to a key escrow problem. Another study by Nguyen proposes an enhanced version of this method for communication between devices in P2P systems [65]. It is based on the idea of a shared secret between two peers, A and B. The whole system has a universal secret (s). Each node A has a public ID A which is used to generate a private key K_A using a localizing function $K_A = L(A, s)$. Alternatively, a PKG can create all these keys and install them on the devices. Then a mating function M is used to create a pair-wise shared key where $M(K_A, B) = M(K_B, A)$. The devices use their pair-wise keys to perform mutual authentication using a Diffie-Hellman approach and generate session key as Figure 3-4 shows.

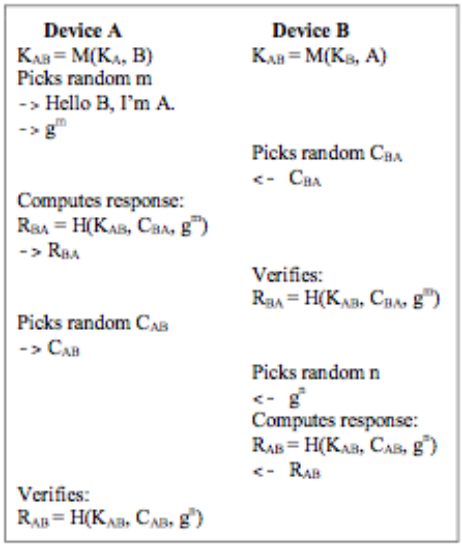

Figure 3-4. IBC node2node Authentication [65]

This method simplifies authentication and limits the initial key installment to one per device. It is also computationally efficient [65]. Cross-domain authentication is also possible through trusted gateway TAG nodes. However, cross-domain authentication may seem insecure if a TAG node masquerades as another node within its domain to take its place in the authentication message exchange. Also, the security of this method depends on the complexity of the identity chosen to be the public key in the first place.

- CL-PKC [54]: or Certificate-less public key cryptography scheme, where a super node and connecting nodes (referred to as the trusted set) provide a distributed key generation center (KGC) based on a master key. This method resolves the key escrow problem. Since it does not require authenticating certificates, this method reduces computation and has a more flexible cryptography workflow. Interoperability is possible with IBC based systems due to similarities. On the other hand, this incurs problems when nodes exit or join the trusted set at any time, leading to the need to update the master key and the distributed partial secrets.

3.5.2. Grid Computing Authentication Methods

Grid Computing (GC) refers to combining computer resources from different administrative domains to achieve a common goal. Both GC and M2C environments are characterized by the on-demand provision of distributed resources, scalability, and multi-tenancy. These characteristics present many security challenges, one of which is authentication. This section presents various authentication methods for GC used in current research:

- Static PWDs [29], [39]: Passwords are the most commonly used method of authentication. It is based on authenticating a user through something he knows. However, passwords are vulnerable to dictionary, guessing, and replay attacks. It also doesn't provide mutual authentication. Reliability of PWDs as means of authentication depends on the secrecy and complexity of PWDs.
- Dynamic PWDs [29], [39]: Dynamic PWDs are either synchronous (based on time or an event) or asynchronous (challenge and response). They provide much more security than static ones. It is based on authenticating a user through something he has, such as a hardware or software authenticator generating One Time PWDs (OTP). Dynamic PWDs eliminate the risk of guessing and replay attacks. However, they still don't provide mutual authentication and credential delegation.
- Kerberos [83], [29], [52]: Kerberos is an authentication mechanism based on symmetric keys. It is a third-party method that includes a trusted centralized key distributed center (KDC) in which the client and the server are registered. KDC contains both an Authentication Service (AS) and a Ticket Granting Service (TGS). The client uses his credentials, usually a username and a PWD to send an authentication request to AS. He receives a Ticket-Granting Ticket (TGT) issued by AS. The client then sends the issued TGT to the TGS, requesting a service ticket. TGS then issues a service ticket, which is used by the client to access the service. Figure 3-5 shows the summary of the messages exchanged in Kerberos to authenticate a client to a web service.

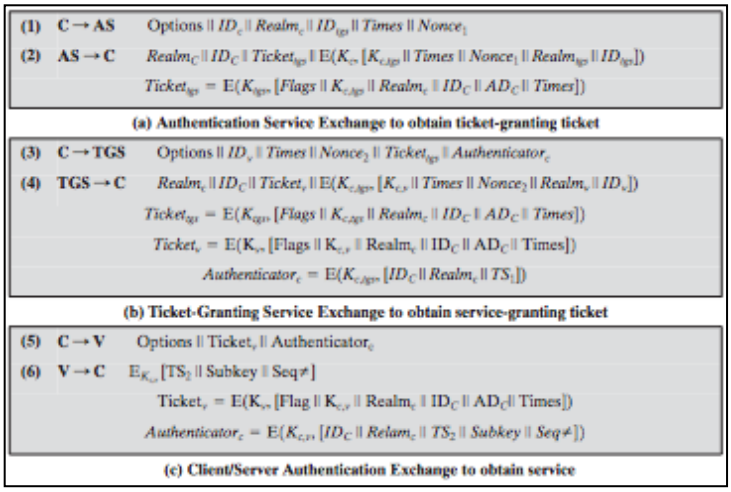

Figure 3-5. Summary of Kerberos 5 Message Exchanges [83]

Version 5 of Kerberos provides mutual authentication. It also allows the use of proxy client tickets and forward addresses for across realm authentication. Still, ticket transfer and confidentiality are

very complex, which makes Kerberos more suitable for localized environments. Secondly, KDC is considered a single point of trust and may be a single point of failure.

- PKI based [29], [39], [14]: Grid Security Infrastructure (GSI) provides a PKI authentication based on X.509 certificates. Each entity (a user or a service) has a pair of PUK and PRK. There is also a trusted third party or a Certificate Authority whose role is to certify and sign all entities certificates. This method suffers from scalability issues and single-point failure. Also, interoperability is limited as it doesn't support integration with systems supporting Kerberos or other schemes.
- Security Proxy Based Trusted Computing (SPTC) [56]: This is an enhancement of PKI based authentication in GSI. This method assumes that clients and servers have employed TPMs, or Trusted Platform Module chips, which store RSA Endorsement Key (EK) and is used along with the owner password to create Storage Root Key (SRK). It also stores a second key, Attestation Identity Key (AIK), which uses hashing to protect the device against unauthorized firmware and software. The architecture of this method includes the client, Root CA to issue identity certificates for users and resources, a domain manager, and a Security Proxy (SP) server which manages user's short-lived proxy credentials. The user first needs to acquire his identity certificate from Root CA and a proxy certificate from SP. In this method, they use OpenSSH CA Server for the Root CA to sign the users' and services' certificates. Two platforms that have AIK certificates and/or identity certificates can mutually authenticate to each other using these credentials. This protocol works as follows:

1- A user x registers with Security Proxy using a shared password through an out-band communications with SP.
2- To use a remote TPM, x obtains a public key of the target TPM, where the matching private key is protected in the TPM, then x sends a request with the public key of the target TPM to SP to create a proxy credential for using the target TPM. SP can encrypt the private key of the created proxy credential using the secret shared with x and the target TPM. Accordingly, the TPM owner allows x to access the target TPM. This method is meant to provide secure communications for the Grid setting of the VO and authenticate users and resources across multi-domains. However, it assumes the installment of the hardware component which is an intrusive and rigid condition unfit for the dynamic SG environment.

- IBC [29], [41], [35] : Identity based cryptography has emerged to overcome the shortcomings of the PKI based scheme, which hinders grid scalability. As explained in Section 3.3.1, IBC defines the user's PUK as the user's ID such as an E-mail or an IP address. Hedayati et al. suggested a similar scheme where PUK is derived from random strings transmitted by the server (with a nonce) to users A & B. Subsequently, A and B can derive their PUKs and reply to the server to prove their ability to create PUKs. The corresponding private key is generated by a PKG using a secret master key and the chosen identity. Then the generated PRKs should be delivered to their respective users over a secure communication channel. This scheme leads to several problems: the key escrow problem, lack of non-repudiation, and difficulty of distrusting PRKs in a large-scale grid environment with various trusted domains.

- CL-PKC [29], [30], [21]: As described in Section 3.3.1, CL-PKC schemes aim to solve the key escrow problem in IBC. CL-PKC provides all the advantages of IBC schemes. However, the KGC's role here is to generate only a part of the PRK of the user. The user can generate his own PRK based on both the partial PRK and a secret value of his choice. Thus, the key escrow problem of IBC is resolved, and the scheme now provides non-repudiation. The mechanism, while considered high security, is usually based on complex bilinear pairings. Farouk et. al. describes a pairing-free CL-PKC scheme [30], where two proxies are used: a User Proxy (UP) and a Resource Proxy (RP). UP is a session manager process that acts on behalf of a user for a limited time. RP is an agent used for inter-domain and intra-domain security operations. The proxies' role is to meet mutual authentication requests between users and resources. A unique node's Distinguished Name (DN) is assigned to each node to facilitate cross domain trust establishment. Each domain has a KGC, and all KGCs form a trust relationship that is built between KGCs before authentication. The proposed scheme has two phases. The cryptographic building blocks of their protocol are elliptic curve PKC and eight variants of SHA hash. This leads to less computational overhead and a one-round authentication protocol run instead of two.

3.5.3. Existing CC Authentication Methods

Most of the current research on CC is geared towards securing CC service provision. In this section, we will examine various authentication methods based on authentication factors discussed in Section 3.2 as well as authentication methods adopted for other distributed systems discussed in Sections 3.3.1 and 3.3.2.

-Multi-level PWD [26]: Dinesha and Agrawal propose a multi-level authentication method for accessing Cloud services, where PWDs are based on three levels of access. The first level is to access the organization, the second level is to access the service, and the third is to define the user privileges for that service. There could be many more levels, depending on the architecture of the system. The PWD is created by generating and adding a piece of the password, which is then passed to the next level. The research provides probabilistic proof that it is much harder to breach this simple system, since the user needs to know passwords for all levels. However, it still suffers from all the shortcomings of PWD methods, including guessing attacks and dictionary attacks. Most importantly, it doesn't provide mutual authentication, or non-repudiation.

- TCG [81]: This method is based on Trusted Computing Group (TCG) Technology, which combines software, special hardware (TPM), and a mediating Trusted Computing Platform (TCP) OS. Users are classified into different access control groups with different permissions for services. A user needs to first register so that he receives ID credentials indicating his access control group. When a user attempts to access CC resource, they provide the acquired access control ID to the server. The server authenticates a user by communicating remotely with the TCP component, to verify the user's identity and relevant information. The TCP component in every user maintains a master secret key for each user, which is used to generate sub-keys for different access roles. These personal keys are registered for approved users of a group of CSPs. Whenever a user attempts to access a service, he needs to generate a session key based on the access control role sub-key and a personal key. This method of security relies on the hardware component being built in by manufacturers, which is an intrusive and rigid method employed in a very dynamic context.

- Cloud Access Manager (CAM) [13]: This method introduces the concept of a Cloud Access Manager (CAM) which is responsible for user authentication and access control for every resource. CAM creates separate access zones with access policies for the resource. A Proxy server accessible from the Internet can communicate with Web servers. Each endpoint registers with the server at a well-known port; the calling endpoint initiates the INVITE/OK/ACK with a port number message exchange endpoint to the same port on the server, which then forwards SIP messages to the registered IP address of the called party. The service messages are then exchanged between the two endpoints via the proxy server on the cloud. The proxy transfers messages from one-endpoint to the other when they cannot reach each other directly. Once the message exchange is complete, the assigned ports on the proxy are released. CAM is entrusted with PRKs used to manage zone

members. These key files are associated with two CAM Access Keys, of which there are two types: an application-level access key, which is used for all members of the defined application, and a member-level access key used to specify a key used for a specific member cloud instance. CAM presents an escrow problem and doesn't allow a non-repudiation provision.

- Dynamic Multi-factor [10]: Banyal et al. introduce a multifactor authentication scheme for three levels of security. In this scheme, a Cloud Administrator (CloudAdmin) is responsible for the Cloud Access Management (CAM) server and cloud computing servers. Internet and web browsers are used to access CAM and CSP to access Cloud resources. Also, authentication credentials (secret key, one-time password, and IMEI number) are exchanged through smart phones and mobile networks as an out-of-band secure channel. A valid email-id is used to send secret and verification codes during the user registration and credential change phases. The proposed method has three phases: the registration phase, the login and authentication phase, and the change authentication credential phase. In the registration phase, a user needs to register at the CAM server by providing a username and PWD, which are registered along with the user's mobile phone and the IMEI number for authentication. Two secret values are sent to the user OOB to mobile number and email to be combined and entered by the user for verification. For the second phase, the user login into CAM system by entering the username and password established during the registration phase. If the login is successful, the user is directed to the homepage for the multi-level Authentication phase. The change authentication credential phase simply allows the user to change the PWD at any time. All Cloud services are classified as low, medium, or high-level. For low-level services, authentication combines passwords and Captcha expressions. For medium level, OTP is combined with Captcha expressions, while high-level services require three factors: OTP, Captcha expressions, and an International Mobile Equipment ID (IMEI) number. This scheme provides dynamic authentication based on the security level, but it doesn't take advantage of the client context (mobility). It doesn't provide mutual authentication. Non-repudiation is guaranteed for med and high-level services.
- InterCloud [17], [57]: This method is aimed at the MC, where multiple clouds need to integrate services for client benefits. It is based on the IdP/SP model, for which SAML is the reference XML-based standard implementation [17]. SAML can establish a trust relationship between entities with different security mechanisms. This method identifies a home cloud as opposed to foreign clouds, each with a different authentication mechanism. To enable the home cloud A to

request resources from the foreign clouds B, C, and D, an authentication task needs to be carried out. Cloud A needs to create accounts on IdPs X and Z as asserting parties, where X is trusted with B and C, whereas IdP Z is trusted with D. Once trust context is established, A gains access to the needed resources, and A has established a federation with other foreign clouds trusted by IdPs X and Z. For A to access B resources, A starts the authentication process by providing its identity to B. The authentication module of cloud B forwards the authentication request to the IdP X initiating an authentication interaction between A and IdP X leading (if successful) to trust establishment relating A to X. The IdP X then sends A's credentials to cloud B. Applying the same concept, Lomotey and Deters suggest an authentication middleware for Mobile consumers of IaaS [57]. The motivation of this method is to resolve the issue of delays in accessing storage Cloud services due to the high data and computational load of HTTP on Mobile clients. The proposed middleware employs the OAuth 2.0 scheme to identify the mobile client and uses security tokens to handle authentication with the IaaS clouds such as AWS S3, Dropbox, and Mega. The scheme enforces additional protection for the security credentials. These IaaS layers require the user to provide credentials such as an access key, a secret access key, signature, and a session id to access the stored files and documents. For mobile clients over wireless links, a credential exchange may fail due to irregular connectivity or suffer from communication latency due to limited bandwidth. The MiLAMob framework allows the mobile user to use the open standard OAuth 2.0 to login. Based on the access token of the registered user, the middleware does the authentication with the IaaS cloud frameworks. The middleware relies on a service level manager to direct the user to Amazon S3, Dropbox, or MEGA. The evaluation of the MiLAMob framework shows high improvements in computation and bandwidth weight in a mobile distributed environment.

- Elliptic Curve Cryptography ID-based method (ECC ID-based): Chen et. al. propose a three-phase ECC ID-based protocol [22]. It is an enhancement of the Yang and Chang ID-based scheme. The three phases are initialization, user registration, and mutual authentication with key agreement phase. The initialization phase is invoked whenever user U_A registers with the authentication server S. S chooses an elliptic curve equation EP (a, b) with order n, and selects a base point P with order n over EP (a, b). S, then, computes its private/public key pair. The PRK is stored, and S chooses three one-way hash functions H1, H2 and H3 used to publish an announcement of the PUK. During the user registration phase, U_A chooses his/her ID_A and

password pw_A and generates a random number b for calculating the b-based password PWB. Then, U_A submits ID_A and PWB to the server S. S creates an authentication key for U_A, and an authenticator B_A to be stored on a smart card and sent to the user over a secure channel. Upon receiving the values, U_A adds random b and his original chosen ID_A and pw_A to verify the authenticator B_A. The mutual authentication with key agreement phase is invoked when U_A requests a service on a remote server S. U_A enters his/her ID_A and PW_A to login to the service, and MS_A, a remote device of U_A, calculates PW_B and B'_A and checks if B'A=BA. If so, it calculates Q and an authentication key, K_{IDA} and chooses a random point R_A with coordinates (x_A, y_A).R is used to compute an anonymous ID of A.U_A then sends a message m with the new key and a timestamp. S verifies the received values and authenticates U_A. In turn, S chooses a random point and performs similar calculations in order to be authenticated to U_A. S also creates a session key to be used for secure communication.

3.6. Chapter Summary

In this chapter, we have presented background information on authentication. We have also analyzed a wide range of authentication methods and solutions producing a classification of authentication solutions in the literature according to the type and the number of factors used to identify an applicant.

The next chapter presents and analyses a real-life use case of MC leading to the derivation of a generic model for a Multi-Cloud (MC) environment. It also explains the threat analysis and general security requirements for MC. It also provides critical analysis of authentication solutions reviewed in chapter 3 solutions in order to identify the knowledge gap in authentication solutions for M2C and deduce the best way forward.

Chapter 4: Problem Identification and Requirements Specification for M2C Authentication

4.1. Chapter Introduction

This chapter describes a real-life application and how it can be used to develop a Multi-Cloud (MC) generic model. It also provides three MC solution scenarios. The description and analysis of these scenarios will allow us to identify issues related to authentication requirements in the MC context.

The later part of this chapter is structured as follows. Section 4.2 provides an analysis of the research problem, including a description of the conditions and requirements of a real-life application to be developed on a MC platform, explanation of the actualization of a single-Cloud Solution, two-Cloud Solution, and a three-cloud solution, step-by-step. Section 4.3 presents the derivation of a generic model for a Multi-Cloud (MC) Solution. Based on the generic model description, a threat analysis of MC is depicted in Section 4.4. Section 4.5 presents observations on the generic model and the threat analysis. Requirements for MC authentication are derived in Section 4.6. Section 4.7 provides an analytical comparison of authentication solutions reviewed in section 3.5 to identify the knowledge gap in authentication solutions for M2C and introduces the best way forward. Finally, section 4.8 introduces high-level ideas for our solution.

The derivation and description of M2C generic model, threat analysis, and identification of security requirements are summarized in the following publication:
Gamlo, Amina H., Ning Zhang, and Omaimah Bamasag. "Mobile cloud computing: security analysis." 2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). IEEE, 2017.

4.2. Problem Analysis and MC Generic Model Derivation

4.2.1. A Real-Life Application: The "NewShop"

In this section, we present the use case of an online retailer company that plans to develop a new Web 2.0 storefront application, referred to as the 'NewShop'. 'NewShop' should accommodate all types of shoppers, including those accessing the store via mobile devices. The application should be stress-tested with real-time data. The company's current user database and product database need to be stored securely to be employed by the application during the testing phase and also by the project upon completion. NewShop's web storefront development, testing, data storage, and web hosting should be done effectively and securely. The IT solution for the above scenario should cater to all the mentioned requirements and conditions.

Considering the above requirements of the retailer company, the use of Cloud Computing (CC) provides an effective solution, as it has many merits. PaaS Clouds provide development tools, eliminating the need to install, configure, and manage tools on individual developer's machines and on the company's site. They also allow speedy product development from the latest source code in Cloud-based storage [5], [37], [61]. Testing in the Cloud can handle the extra burden of Web 2.0 interface interactions with the server. Also, stress testing the new application is easier using the massive number of Virtual Machines (VMs) that are provided in the Cloud environment. In addition, MC affords the possibility of employing different services from different cloud providers as per the user's needs.

Solution scenarios for this application have the following assumptions:

A1: The company has an offline product and user databases.

A2: NewShop Company assigned the development and testing project to a team of developers managed by Admin Bob.

A3: All users are registered with the necessary CSPs. The registration process is secure.

4.2.2. Single Cloud Solution (1C-Shop)

A single Cloud Solution can be employed to meet the requirements of the application described in the prior section. For this requirement, all necessary cloud services are provided through a single CS. This concept will be referred to as the 'One Cloud Shop' scenario (1C-Shop). AWS is one of

many CSPs that may be chosen for implementation. An AWS solution was suggested because Amazon is a leading Public Cloud Provider today with varying services offered.

The following subsections will describe the 1C-Shop solution, presenting the entities interacting and the authentication methods provided for them by AWS, as well as the detailed architecture and operation of AWS services utilized to complete this project.

4.2.2.1. AWS Authentication: Entities and Methods

To understand the authentication provision for this scenario, we need to identify the interacting entities in this setup and the authentication methods and credentials available on this platform.

The interacting entities in this solution are:

1) The CSP offered service, **AWS-S**, which refers to individual AWS services, such as the management console (MC) EC2, S3, and CloudFront.
2) The user, NewShop, is represented by:

- **Admin**: which refers to the team Admin, Bob.
- **User$_j$**: which represents each of the development team members.

3) **User assuming role i**, which points to temporary users, assuming a defined role such as the role of a storefront shopper.

For secure transactions between these entities, AWS offers the following methods and credentials for authentication [93]:

1. Root account login Credentials: The username (a valid email) and password (refered to as PWDroot) are the login credentials for the root account. The password should be chosen, stored securely and changed periodically. They provide access to CSP basic services including MC, discussion forums, and a support center.
2. Identity Access Management (IAM) account credentials, including usernames and passwords (referred to as PWDj for individual user j): IAM accounts are set for individual users by the Admin via the AWS IAM service. Permissions for individual-user accounts can be revoked or modified at any time.

3. AWS Multi-Factor Authentication (MFA): MFA provides an additional level of security for login credentials. It provides OTPs through an MFA software or hardware generator. MFA can be used for the root account (referred to as OTProot) and/or IAM individual-user accounts. (referred to as OTPj for individual user j)
4. Access Keys: Each access key is comprised of (an ID, and a secret). They can be created for the root account (referred to as SAKroot), individual user accounts (referred to as SAKj for individual user j), or IAM roles. An access key is usually used to sign programmatic requests made to services. A sub-type of access key is the temp access key, which has limited life. Temp access keys are issued for temporary user access and are used like regular access keys.
5. Public/Private Key Pairs (referred to as PUK and PRK respectively): These are 2048-bit SSH-2 RSA keys used for services such as EC2 and CloudFront. Some of these services allow creating keys for root accounts only, while others may be created for an IAM user or the root account Admin. Each key pair requires a name. Naturally, the public key is uploaded to AWS, while the private key is securely stored by the user.
6. X.509 Certificates: The certificate contains a 2048-bit RSA public key, with a matching private key (referred to as X509-PUK and X509-PRK respectively). Users are required to use the private key to sign requests to AWS services.
7. Account Identifiers: There are two account identifiers listed on the security credentials page of the root account that can't be changed. Although these identifiers are not used for authentication purposes directly, but they are used to construct Amazon Resource Names (ARNs), which are necessary to refer to resources programmatically to distinguish NewShop resources from other accounts.

For a better understanding of these credentials, Table 4-1 summarizes Who (which entities) Knows What (Credentials).

Table 4-1: Who knows What for AWS Solution

Items \ Entities	Admin	Userj	Service
PWDroot	✓	✗	✗
PWDj	✓	✗	✗
OTProot	✓	✗	✗
OTPj	✗	✓	✗
PUK	✓	✓	✓
PRK	✓	✗	✗
X509-PUK	✓	✓	✓
X509-PRK	✗	✓	✗
SAKroot	✓	✗	✗
SAKj	✗	✓	✓

It should be noted that some credentials listed in Table 4-1 are optional. More information related to each type of credentials, such as the creation, use, and validity of each, is summarized in Table 4-2:

Table 4-2: AWS Solution Credentials

List of Credentials	Type	Required or not	Creation	Use of the credential	Security recommendations	Validity
Root account log-in credentials	Username (Valid email) & password	Required	At the beginning of registration process	Log in to CSP website by Admin	- Change periodically - Don't distribute to individual users	Valid until changed
Individual user log-in credentials	Username & password	Recommend	By Admin for each user	Log in to permitted tools and resources by individual users	- Change periodically	Valid for 90 days, then users are forced to change them
MFA	A mobile app or an MFA device to provide six-digit OTPs	Optional	Set up by Account Admin at any time	Used in conjunction with Log-in credentials (for root account and/or individual users)	-keep device safe from others	One-time use
Access key (ID+ Secret key)	Random strings of characters	Required for individual users	Set up by Admin at any time and downloaded as a CSV file.	Used to sign programmatic requests made by users and to sign commands issued for command line interfaces (CLIs)	- Not recommended for the root account - Should regularly be rotated either by Admin or individual users -Unused keys should be revoked	Until manually revoked
Public/ Private Key Pair for EC2	RSA 2048 bit private and public keys	Required	Set up by Admin or individual users and downloaded as PEM container files	Used to request access to EC2 instances or to create signed URLs for private user content	Recommended to be rotated periodically	Until manually deleted by user or Admin

X 509 certificate	RSA 2048 public key certificate and private key	Required	Set up by Admin and downloaded as PEM container files	Used to sign SOAP-protocol requests and CLI requests to many AWS services	Recommended to be rotated periodically	Until manually made inactive or deleted by Admin
Account IDs	12-digit account ID	Required	Available on Management console right after account creation	Used to construct ARNs necessary to refer to resources programmatically	None	For the lifetime of the account

4.2.2.2. AWS Scenario Architecture and Operation

For a better understanding of the interactions between entities in the 1C-Shop solution, this section describes the architecture of the solution and how entities operate in the frontend and backend. For frontend operation, there are three phases as the SW project progresses:

- Implementation
- build phase
- Testing phase

Backend operations are performed through the NewShop website. Therefore, website hosting will be provided through the same CSP.

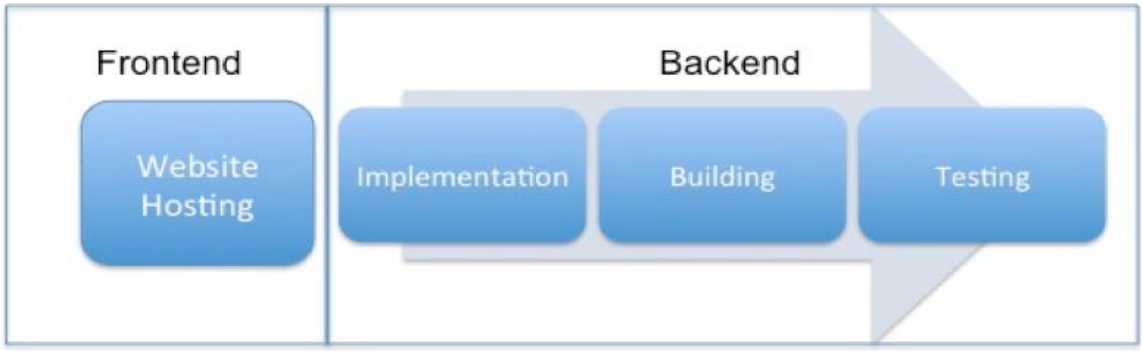

Figure 4-1. Phases of the Solution

Work can progress initially based on the assumption that all team developers are registered. As Figure 4-1 shows, there are four different architectures necessary to fulfil the NewShop's requirements:

The general setup for the 1C-Shop solution depicts an outsourced private cloud. As Figure 4-2 depicts the team enlists the Virtual Private Cloud VPC service, which designates a set of AWS virtual resources to the NewShop's AWS account.

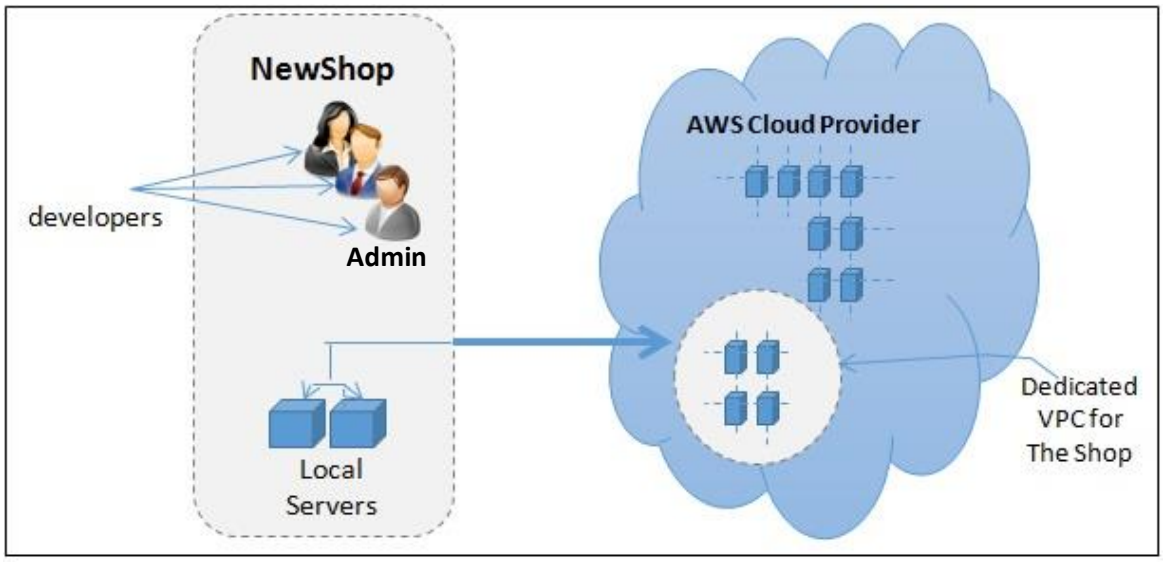

Figure 4-2. A logical view of the AWS solution employed by "NewShop."

An isolated portion of the AWS cloud is created for NewShop to launch EC2 instances with private addresses in the specified range rather than randomly assigned public IP addresses [8]. Network ACLs and security groups provide stringent control over inbound and outbound traffic to instances of the NewShop.

The detailed architecture for each phase enlists different AWS services. As this project progresses from one phase to the next, NewShop's developers can allocate the necessary resources to build the needed architecture for that phase and release unnecessary resources.

(a) <u>Phase 1: Development of the Website</u>

Figure 4-3 shows the architecture of the development phase [96]. The components shown within the VPC subnet are:

- EC2 instances are used for source code repositories and project management tools.

- Amazon Elastic Block Store (EBS), which is a storage service linked to the source code repository EC2 instance.
- S3 object storage service, which allows for storing various types of objects in units called buckets. One S3 instance is used to back-up snapshots of EBS. Another is used to store NewShop's user and product databases.
- An Elastic IP Address is needed to provide a consistent method to statically access EC2 instances.
- RDS is Amazon's relational database service used for data storage for project management tools.

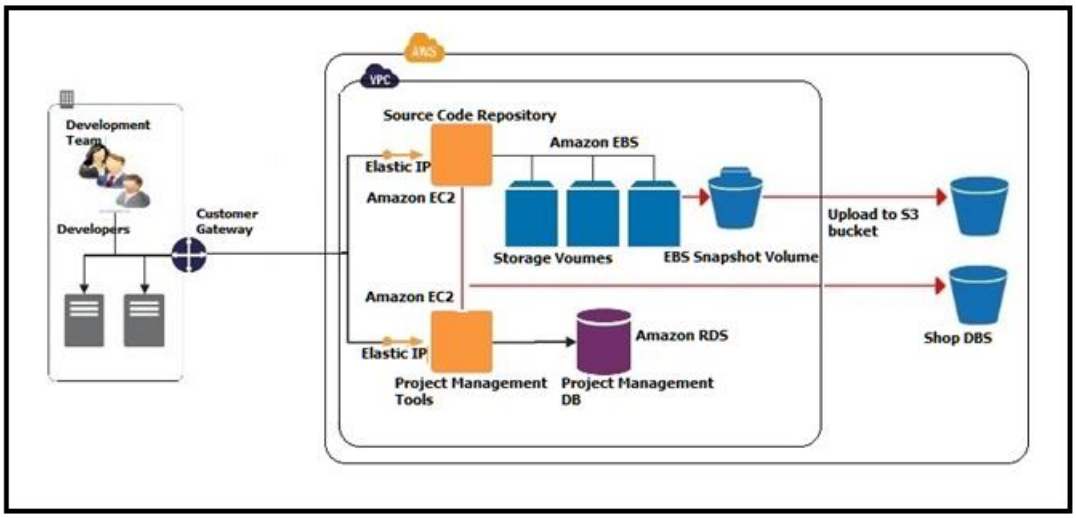

Figure 4-3. The Architecture of "NewShop" AWS solution: Development Phase

For this phase, the admin uses the management console to request the necessary AWS services to deploy the needed architecture for each phase of the project. He also sets up permissions for the team members to access the allocated resources. The development team members use their login credentials to gain access to the management console and resources as per the permissions granted by the Admin.

Through the management console, an S3 bucket called "ShopDBs" is created to migrate the data files. Then the architecture for the development phase must be arrayed as shown in Figure 4-3. EC2 instances are created for source code in the next step. Once the source code repository instance is installed and configured, the team creates an AMI for fast future recreation of that

instance. The repository's data is stored on Amazon Elastic Block Store (EBS), which is attached to the running instance. Additionally, point-in-time snapshots of the EBS repository data volume are created and stored in Amazon S3 to ensure data protection and durability. The Elastic IP Address provides a static access method to code EC2 instances. Additional Amazon EBS volumes may be needed as the code repository grows larger. If required, multiple EC2 instances via the auto-scale service can be created as the project is scaled.

(b) Phase 2: Building the Project

For building the project, the architecture needs to be expanded after the development phase. Figure 4-4 shows the expanded architecture within the VPC.

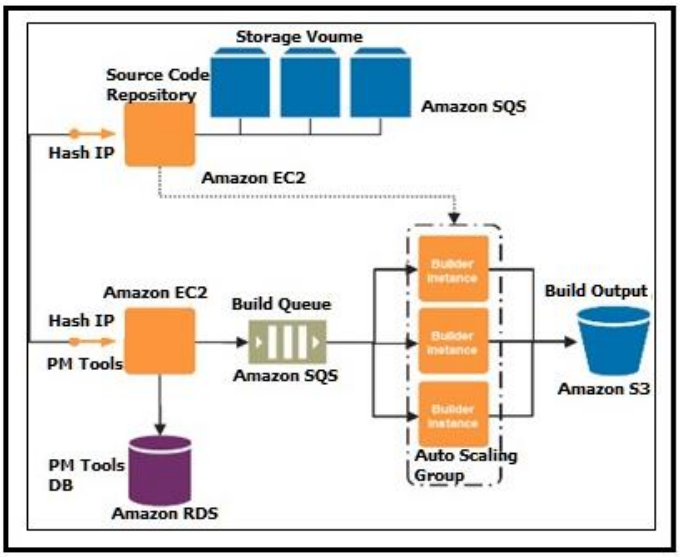

Figure 4-4. Architecture for the Build Phase

The new components necessary for the building phase are:

- EC2 instances to host the build server.
- Auto-Scaling service to adjust the number of EC2 build instances needed.
- Amazon Simple Queue Service (SQS), to manage the multiple build requests polled by the build EC2 instances.
- An S3 instance to store the build output.

For this phase, the team modifies the architecture of the resources, as indicated in Figure 4-4. An EC2 allocation is required to host the build server. For multiple builds within the same day, an

Amazon SQS queue is employed. The number of EC2 build instances is adjusted by the Auto-Scaling service.

(c) Phase 3: Testing the Website

For the testing phase, the same set-up as for the development phase shown in Figure 4-3 can be utilized with EC2, RDS, and S3 services as the main components of the architecture. Different test types require different testing environments. EC2 instances are launched from AMIs configured with test environment requirements. Test datasets are implemented as Amazon RDS instances loaded from stored snapshots. Stress testing is accomplished by large enough RDS datasets and the availability of multiple required EC2 test instances. EC2 testing images are distributed over different AWS regions to gain a better understanding of the end-user experience and response times. Fault Tolerance testing is easily accomplished using the AWS Management Console to take down some components, such as EC2 instances, to test the architecture. After testing, NewShop's web storefront is ready for deployment on AWS.

(d) Phase 4: Hosting the NewShop's Website

The final phase of the project after development is building and testing. For this phase, the architecture employed is a basic 3-tier web application architecture, as presented in Figure 4-5.

The component services of this architecture are:

- Amazon Route 53 DNS service, which routes the network traffic to the AWS components.
- Amazon Cloud Front distributes content (dynamic, static, streaming, and interactive) using a global network of edge locations to end users.
- Amazon S3, for static content and backups.
- Amazon Elastic Load Balancer, which facilitates the distribution of computing load.
- Amazon EC2 instances used for the creation of web servers and application servers.
- Amazon ElastiCache, for in-memory application caches.
- Amazon RDS, for storing the application databases.

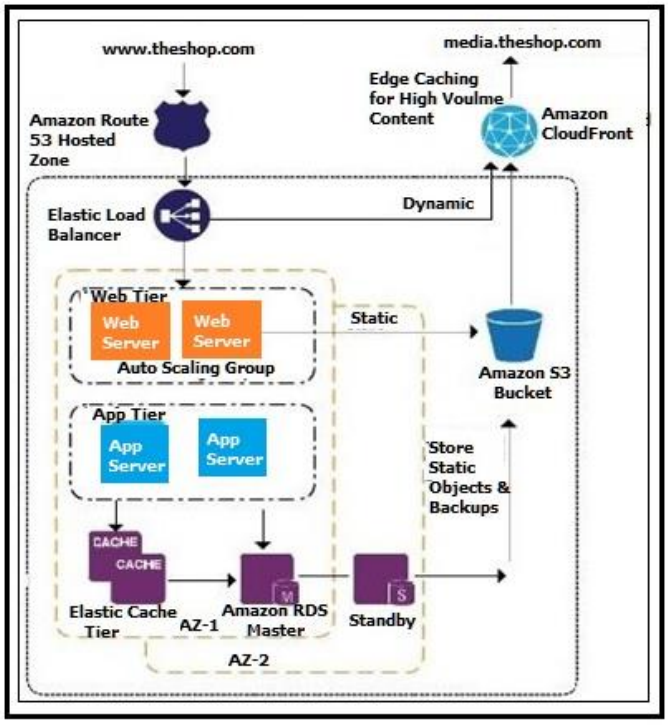

Figure 4-5. AWS Architecture for Web Hosting Solution [86]

In this phase, Admin allocates the necessary components on AWS. NewShop's databases are transferred from the S3 bucket allocated in phase 1 of the project, to Amazon RDS to be operational. Amazon CloudFront delivers dynamic and static content for NewShop, using a global network of edge locations. The NewShop's online customers will use the hosted website. These users will assume a role defined as "shopper" to allow suitable access to the website. Their requests are automatically routed to the nearest edge location to ensure the best performance in content delivery. The AWS DNS service (Route 53) is provisioned to resolve requests for www.theshop.com to the Elastic Load Balancer. The Load Balancer receives HTTP requests and distributes them among web servers running on EC2 instances across multiple zones. The number of web servers within the auto scale group is scaled up or down in response to incoming traffic. EC2 instances are also created to host app servers, where the NewShop's application is hosted. Elastic IP addresses are assigned for EC2 instances and services requiring static endpoints, such as central file servers, and load balancers. Amazon ElastiCache is used to reduce the load on services and improve performance. An S3 bucket is allocated to store static data and backups.

4.2.3. Two Clouds Solution (2C-Shop)

In this solution, development, building, testing, and hosting for the new web storefront are deployed on AWS CSP as in the 1C-Shop scenario. However, data storage services are provided by a different CSP X. This scenario will be referred to as 2C-Shop. Assumptions A2 and A3 from Section 4.2.1 are applicable in this setup. Additionally, we assume:

A4: The company has product and user databases, which are hosted by CSP X.

The following subsections will analyze the 2C-Shop solution, describing the interacting entities, the authentication methods, and the architecture and operation of services utilized for project completion.

4.2.3.1. 2C-Shop Authentication: Entities and Methods

In this section, interacting entities are identified along with the authentication methods and credentials to be used.

The interacting entities in this solution are:

1) AWS is represented by the services it offers. **AWS-S** refers to individual AWS services, such as the management console (MC), EC2, S3, and CloudFront.
2) CSP X is represented by the services it offers. **CSPx-S** refers to an individual service.
3) The user, NewShop, is represented by:
   - **Admin**, identified as the team Administrator (Bob)
   - **User$_j$**, who are the development team members
4) **User assuming role i**, which refers to temporary users, who have a well-defined role such as the role of a storefront shopper

The credentials structure for this scenario is similar to the one presented in Section 4.2.2.1. Table 4-1 and Table 4-2 summarizes the access control details for the users accessing the resources provided by AWS. The mandatory credentials required to access CSP X resources should augment the previously listed credentials (Table 4-1). Table 4-3 summarizes the additional credentials:

Table 4-3: 2C-Shop Solution Additional Credentials

List of Credentials	Type	Creation	Use of the credential	Validity
Root account log-in credentials for CSPx	Username (Valid email) & password	At the beginning of registration process	Log in to CSPx website by Admin	Valid until changed
Individual user log-in credentials for CSPx	Username & password	By Admin for each user	Log in to permitted tools and resources by individual users	Valid for 90 days, then users are forced to change them
MFA for CSP X	SW or HW device to provide six-digit authentication codes (OTP)	Set up by Admin at any time	Used in conjunction with Log-in credentials (for root account and/or individual users)	One-time use
Access key (ID & secret key)	Random strings of characters	Set up by Admin at any time and downloaded as a CSV file.	Used to sign programmatic requests made by users.	Until manually revoked

For a better understanding of these credentials, Table 4-4 summarizes Who (which entities) know What (credentials). All credential items in Table 4-4 refer to those used to access CSPx and the services it provides to the user. Users' X509 certificates and matching private keys are used for both CSPs.

Table 4-4: Who knows What for 2C-Shop Solution

Entity / Item	Admin	User j	User assuming role i	CSP X
PWDroot for CSP X	✓	×	×	✓
PWDj for CSP X	✓	×	×	✓
OTProot for CSP X	✓	×	×	×
OTPj for CSP X	×	✓	×	×
X509-PUK for CSP X	✓	✓	×	✓
X509-PRK for CSP X	✓	✓	×	×
SAKroot for CSP X	✓	×	×	×
SAKj for CSP X	×	✓	×	×

The **admin** holds the root account credentials for both CSPs. The MFA producing OTPs are optional for the root accounts and individual user accounts. The **admin** holds ownership of the OTP device for the root account, while each user owns the OTP device for their account. An access key is created and is owned by each user and for each role defined by the **admin**. Secret access keys are used to sign REST API requests. The private key matching the public key of the X.509 certificate created with AWS can be used to sign requests for services offered by CSPx.

4.2.3.2. 2C-Shop Architecture and Operation

The general architecture for this scenario is shown in Figure 4-6. This configuration is similar to the one presented in Figure 4-2 in Section 4.2.2.2. It depicts how NewShop is utilizing the services of AWS services as well as another CSP X.

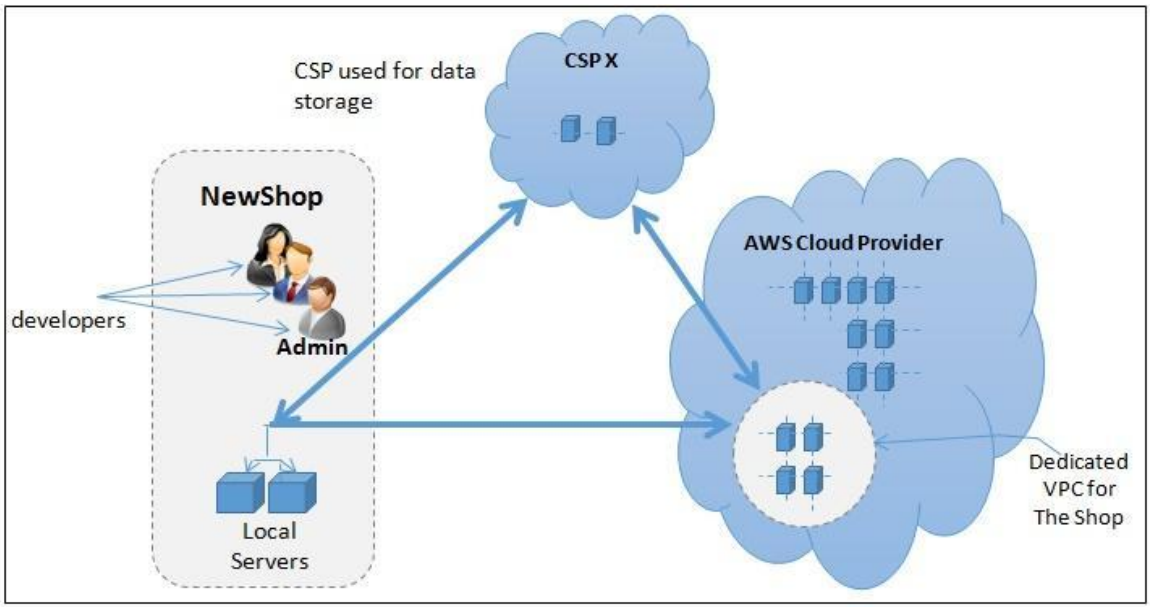

Figure 4-6. General Architecture of 2C-Shop Solution

Assuming all users are registered for AWS and CSPx services, the team of developers can start working on the four phases of this project: Development of the website, Building the project, Testing the storefront, and hosting the website. All four phases employ AWS services. Throughout the project, NewShop's product and customer databases are stored on the storage service offered by a different CSP X.

(a) Phase 1: Development of the Website

For this phase, NewShop employs an architecture of AWS services, as shown in Figure 4-3. The admin requests the necessary services, which are EC2, EBS, S3, RDS, and the Elastic IP address. They operate in the manner described in the Section 4.2.2.2 (Phase1).

(b) **Phase 2: Building the Project**

For building the project, the architecture is expanded, as illustrated in Figure 4-4. It requires EC2 for the build server, the Auto Scaling service, SQS, and S3 instance to store the build output. These components operate as described in the Section 4.2.2.2 (Phase2).

(c) **Phase 3: Testing the Website**

For this phase, the architecture is shown in Figure 4-7.

The components of this solution include all AWS components described in Section 4.2.3. An additional perimeter is depicted in Figure 4-7 for the resources implemented through the second CSP X. Since Cloud X is only employed for storing NewShop's databases, it shows only one storage component equivalent in function to the AWS S3 service.

EC2 instances are launched and configured with various test environment requirements. Test datasets are stored in the storage component provided by CSPx. EC2 instances need to request this data when needed from CSPx. Stress testing is accomplished through large datasets and by launching as many necessary EC2 test instances as necessary. After the testing phase is complete, NewShop's web storefront is ready for deployment on AWS.

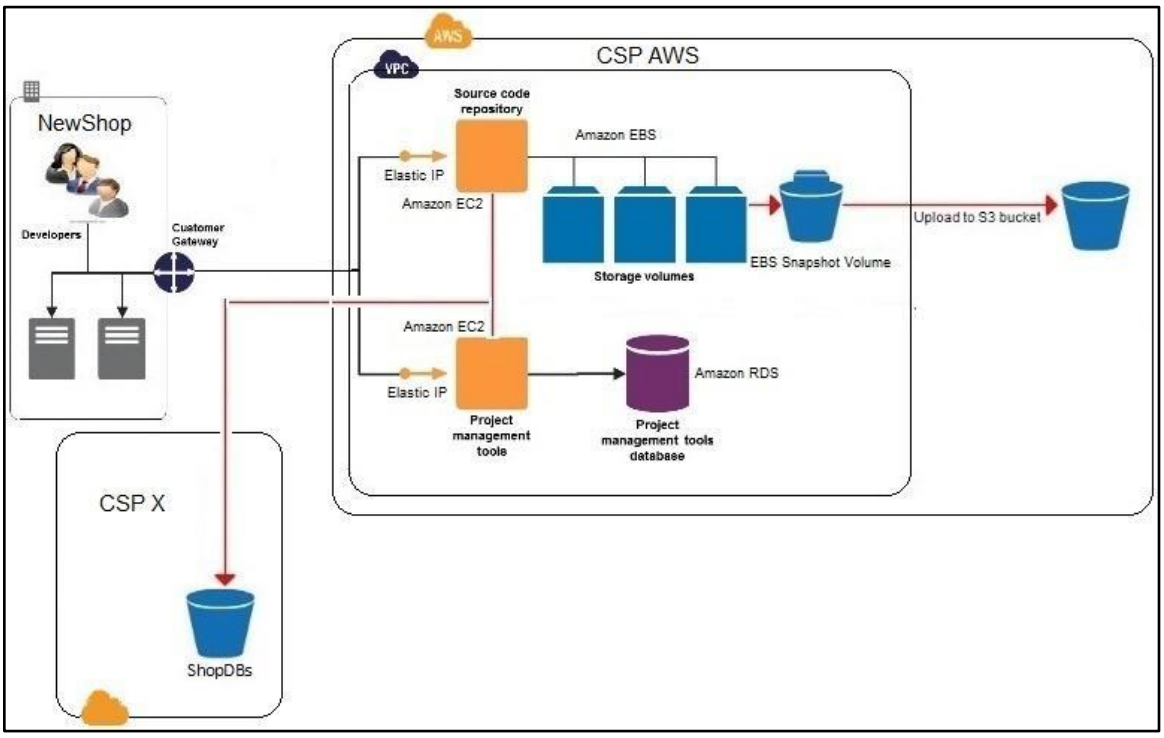

Figure 4-7. Architecture of 2C-Shop (Phase 3)

(d) Phase 4: Hosting NewShop Website

For the final phase of this project, the architecture used is a basic 3-tier web application architecture, as shown in Figure 4-5.

After allocating all the necessary components on AWS, the operation of these components is described in Section 3.2.2.2 (Phase 4).

4.2.3. Three Cloud Solution (3C-Shop)

In this solution, NewShop hires three CSPs for the project. AWS, as in the 1C-Shop and 2C-Shop solutions, is still chosen for the development, building, and testing of the new web storefront. Also, data storage services are provided by CSPx. In addition, hosting the NewShop's website will be provided by a third-party provider CSPy. Hence, this scenario will be referred to as 3C-Shop. Assumptions A2 and A3 from Section 4.2.2 and assumption A4 from Section 4.2.3 still apply in this situation.

The following subsections will describe the interacting entities, the authentication methods, the architecture and operation of services utilized to complete the project.

4.2.3.1. 3C-Shop Authentication: Entities and Methods

In this section, the interacting entities, the authentication methods, and credentials are identified.

The interacting entities in this solution are:

a. AWS, represented by any of the services it offers, **AWS-S**.

b. CSPx, represented by any of the services it offers, **CSPx-S**.

c. CSPy, represented by any of the services it offers, **CSPy-S**.

d. The users, which include the following NewShop's workforces represented by:

- **Admin**, which refers to the team Admin, Bob.
- **User$_j$**, which represents each of the development team members.
- **User assuming role i**, which refers to temporary users, with clearly defined roles, such as the role of a storefront shopper.
- **User assuming role i**, which refers to temporary users, with specific defined roles, such as the role of a storefront shopper.

The credential structure for this scenario is very similar to that presented for 1C-Shop and 2C-Shop. For the users to access the resources provided by AWS, the credentials are summarized in Table 4-2. Additional credentials necessary to access CSP X are mentioned in Table 4-4. Table 4-5 summarizes the additional credentials:

Table 4-5: 3C-Shop Solution Additional Credentials

List of Credentials	Type	Creation	Use of the credential	Validity
Root account log-in credentials for CSPy	Username (Valid email) & password	At the beginning of registration process	Log in to CSP website by Admin	Valid until changed
Individual user log-in credentials for CSPy	Username & password	By Admin for each user	Log in to permitted tools and resources by individual users	Valid for 90 days, then users are forced to change them
MFA for CSPy	SW or HW device to provide six-digit authentication codes (OTP)	Set up by Admin at any time	Used in conjunction with Log-in credentials (for root account and/or individual users)	One-time use
Access key (ID & secret key)	Random strings of characters	Set up by Admin at any time and downloaded as a CSV file.	Used to sign programmatic requests made by users for REST APIs.	Until manually revoked

For a better understanding of these credentials, Table 4-6 summarizes Who (which entities) know What (credentials). All items in Table 4-6 refer to credentials used to access services offered by CSPy. A X509 certificate and matching private key is used for all three CSPs.

Table 4-6: Who knows What for 3C-Shop Solution

Entity / Item	Admin	User j	User assuming role i	CSP Y
Y-PWDroot	✓	✗	✗	✓
Y-PWDj	✓	✗	✗	✓
Y-OTProot	✓	✗	✗	✗
Y-OTPj	✗	✓	✗	✗
X509-PUK	✓	✓	✗	✓
X509-PRK	✓	✗	✗	✗
SAKroot	✓	✗	✗	✗
SAKj	✗	✓	✗	✗

The **admin** holds the root account credentials for all three CSPs. The MFA that produces OTPs is optional for the root account and for individual user accounts for all three CSPs. The **admin** holds ownership of the OTP device for the root accounts, while each User owns the OTP device for his account. An access key is preserved and owned by each individual user and for each role defined by the **Admin**. Secret access keys are used to sign REST API requests. For signing SOAP requests, the private key matching the public key of the X.509 certificate is created for AWS can be used.

4.2.3.2. 3C-Shop Architecture and Operation

The general architecture of this scenario is illustrated in Figure 4-8. This configuration is similar to that presented in Figures 4-2 and 4-6 It shows how NewShop utilizes the services of AWS services as well as those of two other CSPs X and Y.

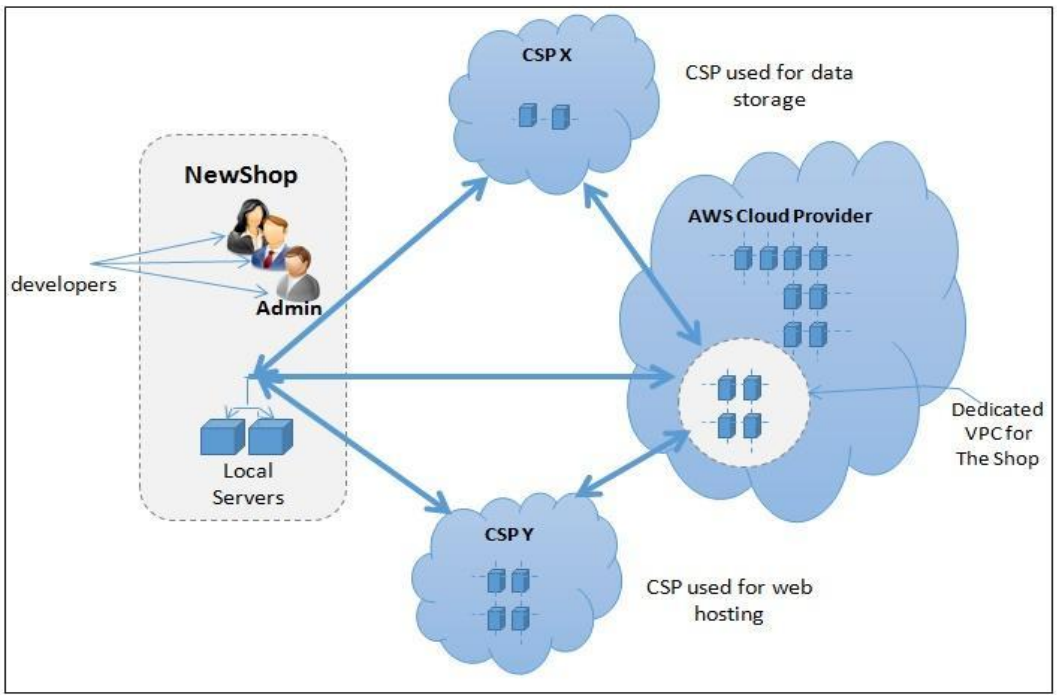

Figure 4-8. Architecture of 3C-Shop solution

After registration, the team of developers can work on the four phases of this project: phase1 (Development of the website), phase2 (building the project), phase3 (testing the storefront), and phase4 (hosting the website). Phases 1-3 employ AWS services. Phase 4 (hosting the NewShop's website) will be accomplished through CSPy services. Throughout the project, NewShop's product and customer databases are stored on a different CSPx storage service.

(a) Phase 1: Development of the Website

For this phase, NewShop employs the architecture of AWS services as shown in Figure 4-3. The admin requests the necessary AWS services. They operate as per the description provided in Figure 4-6.

(b) Phase 2: Building the Project

For building the project, the architecture is expanded as shown in Figure 4-4. The allocated services operate as described in Figure 4-4.

(c) Phase 3: Testing the Website

For this phase, the architecture is shown in Figure 4-7. The operation of this phase is the same as that described in Figure 4-7.

(d) Phase 4: Hosting NewShop Website

For this final phase of this project, the architecture is a basic 3-tier web application architecture, as shown in Figure 4-5. NewShop's project code is relocated and deployed on CSPy resources. All compute, storage, and supporting services are administered by CSPy rather than AWS. The operation of these components is similar to the one described in the Section 4.2.2 (Phase 4).

4.3. Multi-Cloud Generic Model (MC-Model)

Based on the previous solution scenarios: 1C-Shop (Section 4.2.2), 2C-Shop (Section 4.2.3), 3C-Shop (Section 4.2.4), the current study describes a generic abstract model for multi-cloud solutions. The abstract model can serve as a basis for designing an authentication solution for MC. The model may be referred to as the MC-Model. This general scenario requires the collaboration of services offered by two or more CSPs.

This section is structured as follows. Section 4.3.1 will describe the general architecture of the model. In Section 4.3.2, we identify the interacting entities in the model. The credential structure for the model is described in Section 4.3.3. Section 4.3.4 identifies the MC-Model workflow. Section 4.3.5. concludes with a detailed description of the interactions and communication messages between the interacting entities.

4.3.1. MC-Model Architecture

This section describes the general architecture of a Generic MC-Model solution. As Figure 4-9. depicts, a user's request in a multi-cloud environment is fulfilled by multiple cloud providers. Hence, the multi-cloud (MC) System Model can be defined at a given time t by:

$MC = \langle U, P, S, G \rangle^t$ *(4-1)*

where U = ⟨u_1, u_2, u_3... ⟩ is the set of users, P = ⟨ p_1, p_2, ..., p_n ⟩ is the set of cloud service providers, S = ⟨s_1, s_2... ⟩ is the set of cloud services, G = ⟨g_1, g_2... ⟩ is the set of cloud groups, each providing services to one user. Each user u acquires multiple services s_a, s_b … , which are delivered by multiple CSPs represented by a cloud group g where g_i = p_l, p_m, ..., p_n, .

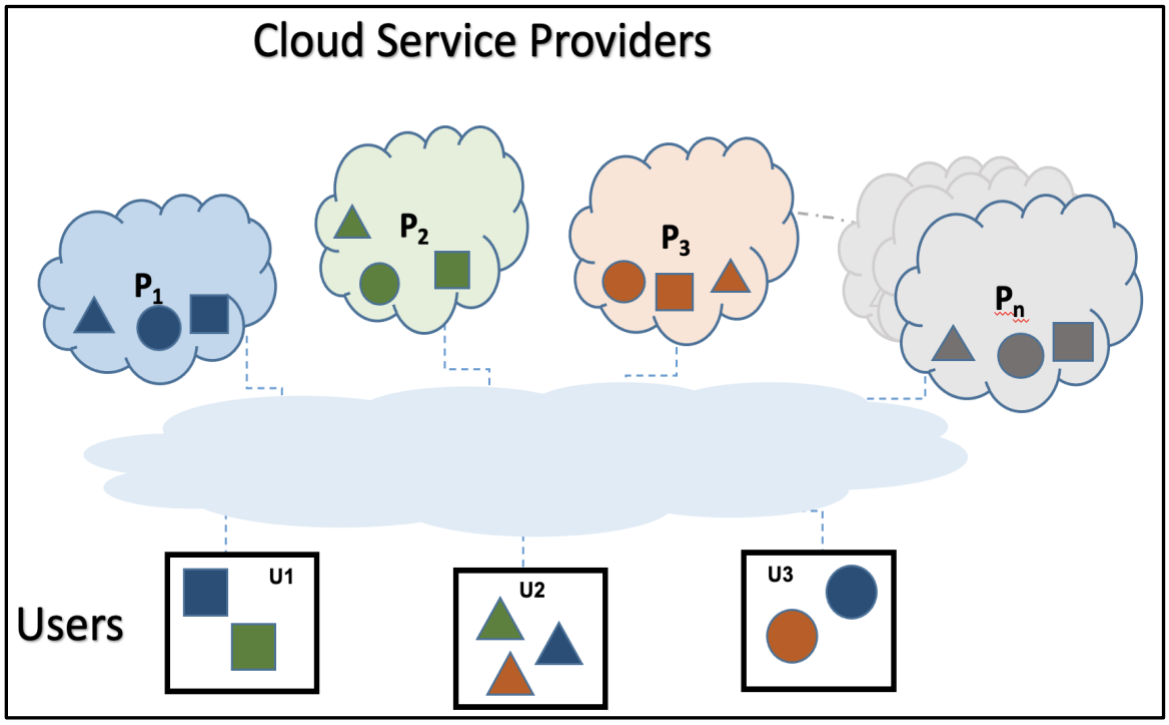

Figure 4-9. Logical view of Multi-Cloud Generic Model

This setup shows multiple security perimeters. Each user u implements a security perimeter, represented in the graph by a black square. Other perimeters are set around the assets of each provider p, shown as a colored cloud shape. Each provider offers various services, represented in the graph as unified color shapes within the provider cloud. Each user deploys instances of various services offered by different providers. Communications links connecting users to providers go through the Internet.

4.3.2. Entities

The interacting entities of the model are:

1. User **u** which represents an organization or a single user.

2. For **n** providers denoted by $\mathbf{p_i}$, each p offers various services, each of which can be denoted as $\mathbf{s_y\text{-}p_i}$ which refers to an individual service offered by one provider $\mathbf{p_i}$.

So, $\mathbf{s_2\text{-}p_1}$ refers to service $\mathbf{s_2}$ offered by $\mathbf{p_1}$.

4.3.3. Credentials Structure

To understand the authentication provision for this generic scenario, the security credentials need to be identified, which are as follows:

1. **C1**: The username and password are the login credentials for the root account. It is denoted by PWD_{root}. All commissioned resources can be accessed through the root account.

2. **C2**: Individual user account credentials, PWD_u, which are usually set by an admin via the root account.

3. **C3**: On-time password, which is denoted by OTP. OTPs are provided by a Multi-factor Authentication generator (MFA) to provide an additional level of security for login credentials. It can be used for the root account, $OTP_{root,}$ and/or the individual-user accounts, OTP_u. OTP generator can be implemented as a hardware device or a software app.

4. **C4**: Non-certified public/private key pairs, denoted by PUK and PRK, can be created for an individual user (PRK_U with matching PUK_U) or for the root account (PRK_{root} with matching PUK_{root}). They can be created by the CSP, or by a third-party. An example of a third-party is *ssh-keygen* which is the tool provided with the standard OpenSSH installation. If key pairs are created by a third party, PUK needs to be uploaded to the CSP. This PUK is not certified and is saved by both CSP and the user. The PRK_U is used to encrypt user requests with login information to access services such as computing instances. At the receiving end, a cloud service uses the matching key PUK_U to decrypt the request and login information. Access is granted only if the login information matches the one stored in the database.

5. **C5**: Certified public key (X509-PUK) with matching private key (X509-PRK). A user u signs requests issued to service s with his X509-PRK.

6. **C6**: Secret Access key. It is denoted by SAK. Secret access keys are created for the account Admin (SAK_{root}) or for an individual user (SAK_U). The SAK is a shared secret between u and p.

It is used to sign programmatic requests to cloud services. When a user application issues a request for a cloud service, SAK is used to calculate a SHA based HMAC signature.

Summary of the above credentials is listed in Table 4-7. It shows the type of each credential, whether it is required or not, security recommendations, and validity for each.

Table 4-7: MC-Model Credentials

List of Credentials	Type	Required or not	Security recommendations	Validity
PWD_{root}	Password	Required	- Change periodically - Don't distribute to individual users	Valid until changed
PWD_{u}	Password	Recommend-ed	No extra recommendations	Valid for 90 days, then users are forced to change them
OTP_{root}	A mobile app or an MFA device to provide six-digit authentication codes	Optional	No extra recommendations	On- time use
OTP_{u}	A mobile app or an MFA device to provide six-digit authentication codes	Optional	No extra recommendation	One-time use

PUK/PRK	Un-certified public and private key pair	Required	Recommended to be rotated periodically	Until manually deleted by user or Admin
X.509-PUK/ X.509-PRK	X.509 certified public key and private key	Required	Recommended to be rotated periodically	Until manually made inactive or deleted by Admin
SAK	Secret Access Key which is a random string of characters	Required	Recommended to be rotated periodically	Until manually revoked

Table 4-8 provides a summary of all credential items and interacting entities showing Who knows What for the MC-Model model.

Table 4-8 Who knows What for MC Model

Item	Admin	U	P
PWD_{root}	✓	✗	✓
PWDu	✗	✓	✓
OTP_{root}	✓	✗	✗
OTPu	✗	✓	✗
PUK_{root}	✓	✓	✓
PRK_{root}	✓	✗	✗

PUKu	✓	✓	✓
PRKu	✗	✓	✗
X509-PUK	✓	✓	✓
X509-PRK	✗	✓	✗
SAK_{root}	✓	✗	✗
SAKu	✗	✓	✗

4.3.4. MC-Model Workflow

For better understanding of the message sequence exchanged between the entities of MC, a simplified example run derived from the 2C-Shop in Section 4.2 is presented (Testing phase). For 2C-Shop, two CSPs collaborate to finish a test case of the implemented online storefront. Accordingly, MC = ⟨U, P, S, G⟩ where,

the set of users U = ⟨Admin, user1, use2, user3⟩,

the set of cloud providers P = ⟨AWS, CSP X⟩,

the set of cloud services S = ⟨EC2, S3, RDS, EBS, AWS.M, X.M, $X.S_1$⟩,

and there is one cloud group g , where g_1 = P.

Figure 4-10 shows the component resources and illustrates the sequence of messages exchanged to accomplish a test case for the collaboration of two CSPs.

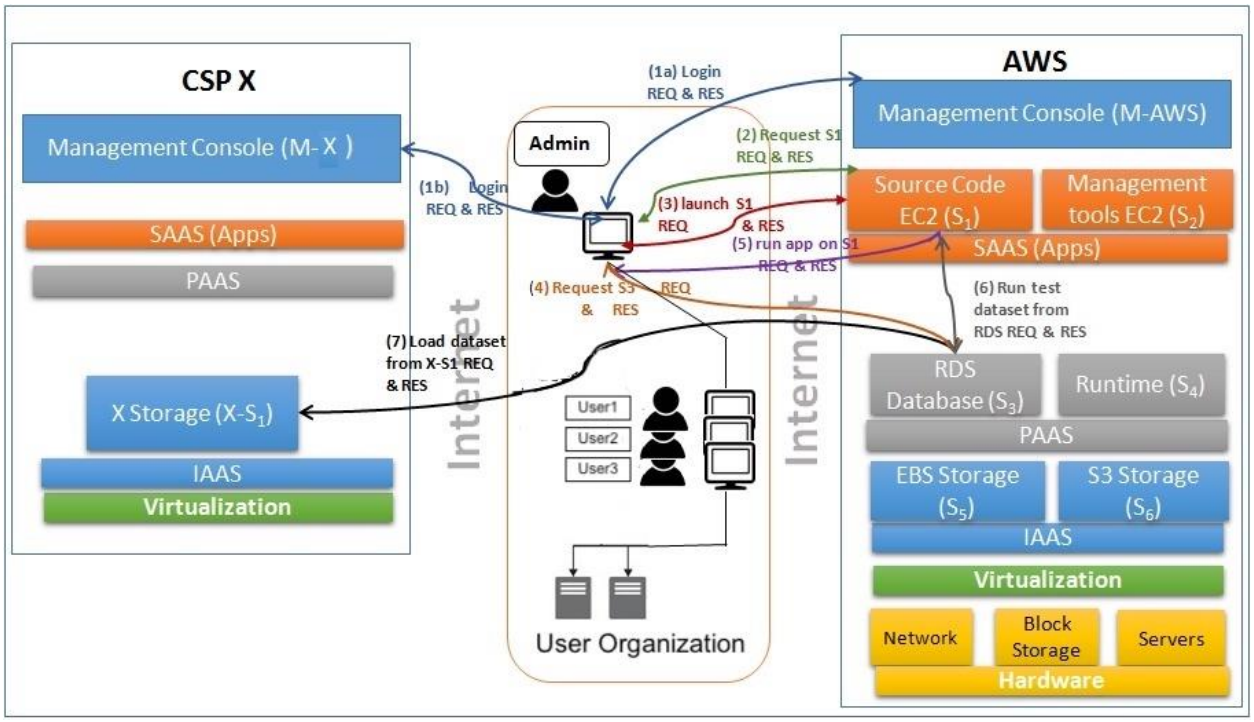

Figure 4-10. Simplified Example Run from 2C-Shop Solution

For this example, the step-by-step workflow, for registered user service utilization is given below:

(1) The user employs credentials created during registration to login to AWS and CSP_x. The credentials usually include PWD and OTP.

(2) The user (Admin/other) communicates with the management console to request EC2 service, s_1-AWS. The request specifies the type of compute instance required, among other attributes.

(3) The user requests to access the allocated service s_1 by requesting to launch an instance of EC2 to accomplish the task.

(4) The user (Admin/other) communicates with the management console to request RDS service, S_3-AWS.

(5) The user issues another access request to the allocated service s_1 by requesting to run an app on the EC2 instance.

(6) The app running on s_1 requests to access another service S3 (RDS) of the same provider, AWS.

(7) As processing continues, s_1 requests access to storage services provided by a different provider, $S_1.X$.

4.3.5. Interactions between Entities and Message Types

Figure 4-11. shows, interactions between entities in MC-Model. It can be further classified into the following two types:

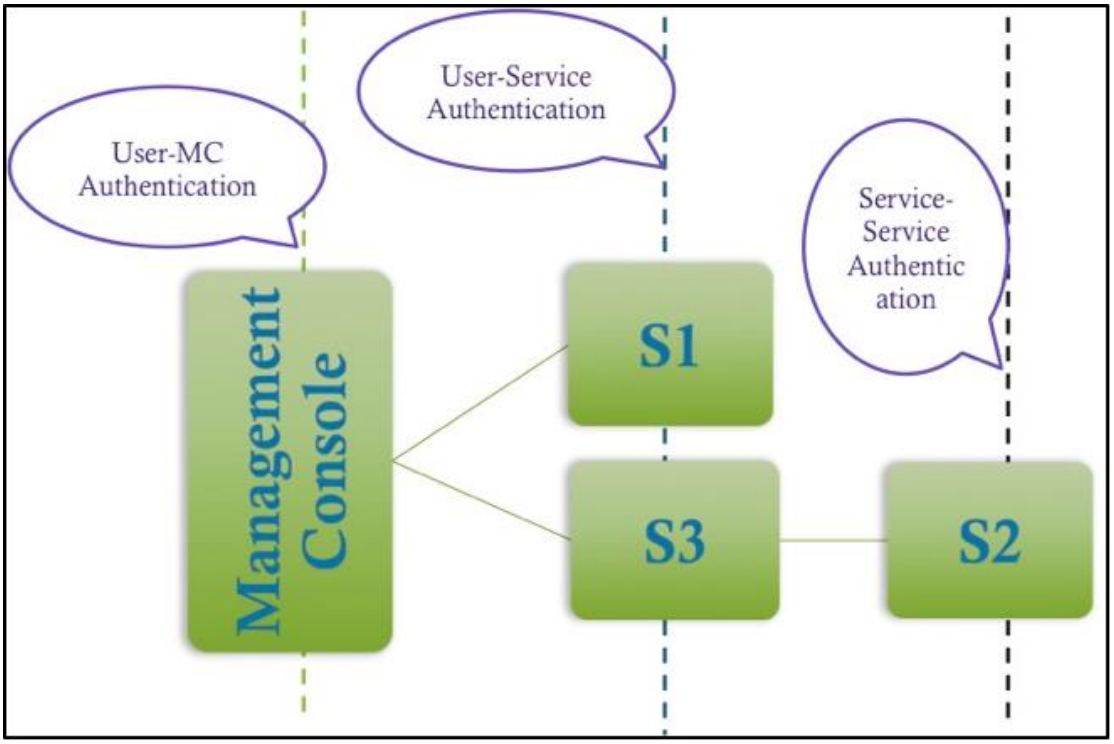

Figure 4-11. MC-Model Interactions

1. **User to Service (U-to-S) interactions:** This type includes interactions between a user and services offered through the management console of a provider, which is represented by green dotted line, and between a user and all other types of services, which is denoted by a blue dotted line**.** Registration, credential exchange, and users requesting services from providers are examples of User-to-MC interactions. For this type of interaction, authentication may be provided through passwords, multiple-factor authentication, and/or OTPs. For requesting other types of services, authentication may be provided through a public/private key pair, X509 certificates, or secret keys depending on the service type.

2. **Cloud Service to another Cloud Service (S-to-S)**: This type of interaction is represented by a black dotted line. Authentication should be provided for both services to safeguard the user's

data and processes as well as those of other users of both providers. There are two distinct types of this interaction:

a. Service-to-Service where both interacting services are offered by the same CSP.

b. Service S_a offered by P_x to another Service S_b on P_y.

Each of these interactions requires communication messages to be sent back and forth between the entities. For each of these interactions, the types of messages communicated between the entities are identified. Generally, there are two types of messages: request (*REQ*) and response (*RES*). Both types of messages may contain supporting data.

(i) **Login**

After initial registration and credential exchange, a user **u** needs to login to **provider p** to start using cloud services. The login process is User-to-S interaction. To facilitate the registration process, the messages proceed the following way:

The messages communicated for **U-to-S** interaction, where **s** in this interaction is usually the management console of p **(M-p)**, are as follows:

- REQ_u: A request message from the **u** with the username and PWD.
- RES_p*:* If PWD is correct, **s** requests OTP. Otherwise, the request is denied.
- RES_u*:* A response message is sent with the generated OTP.
- RES_p*:* If the OTP is correct, a response message is sent to allow access to **M-p.** Otherwise, the request is denied.

(ii) **Requesting Resources**

This includes requesting a resource of a specific service type and setting up specifications for the requested service. An example of this is when **u** request the creation of a storage bucket, and specify the attributes of the resource, such as the ID and size of the storage bucket.

U-to-S interaction messages for requesting a storage bucket are as follows:

- REQ_u: A request message from **u** is sent to create the resource. The request contains supporting data, including resource name, ID, size, access permissions for other users (in case **u** is an **admin**) and other necessary information.

- *RES_p:* If **u** has permission to allocate resources of this type, a response message from **M-p** confirms the creation of the instance with the prescribed properties. Otherwise, the response message denies the request.

(iii) **User Accessing Service**

After a service is requested and deployed, the **user u** needs to access an allocated service **s**. An example of this is a **user** requesting to retrieve an object stored in a storage bucket. This interaction is between **u** and **s-p_x**, where **s** is the storage service offered by **CSP_X**.

Interaction messages would be different depending on the service to be accessed. For instance, the interaction messages for **u** requesting to retrieve an object from a storage bucket are as follows:

- *REQ_u:* A request message from **u** to retrieve an object stored in a storage resource. The request contains supporting data, including resource Name, ID, and object ID.

- *RES_S:* If the **u** has permission to access the requested object, a response message from **s** delivers the requested object. Otherwise, the response message denies the request.

Similarly, the messages facilitating the request of **u** to launch a compute instance are:

- *REQ_U*: A request message is sent from **u** to run an application on a working compute instance. It contains the resource name, information about the application to be run and other supporting data.

- *RES_s:* If the **u** has permission to utilize **s**, a response message from **s** confirms the requested operation with a link to the application. Otherwise, the response message denies the request.

(iv) **Service Accessing Service**

When a task performed by service **s-p_x** requires accessing another service on the same domain, it is a **S-to-S** interaction. An example of this would be when S_a needs to access data stored in the database service S_e. The messages of this interaction would go as follows:

- *REQ_{Sa}*: A request message from **S_a-P_x** to retrieve data stored in Database **S_e-P_x**. The request contains supporting data, including Database Name, ID, and query among other necessary information.

- *RES_{Se}:* If the requesting **S_a-P_x** has permission to access the data, a response message from **S_e-P_x** delivers the requested data. Otherwise, the response message denies the request.

Alternatively, if a task performed by **s-P_x** requires accessing a resource deployed on another CSP_y, it is also considered as a **S-to-S** interaction. An example of this would be when S_a-P_x needs to access an object stored in S_b-P_y. The messages of this interaction would flow as follows:

- *REQ_{Sa}*: A request message from **S_a-CSP_x** is sent to retrieve an object stored in a storage resource. The request contains supporting data, including resource Name, ID, and object ID among other necessary information.

- *RES_{Sb}:* If the requesting **S_a** has permission to access the object, a response message from **S_b** delivers the requested object. Otherwise, the response message denies the request.

This type of interaction can also be recognized as a **P-to-P** interaction, which is the focus of multi-cloud (MC) solutions. Further, MC brings serious concerns of security and privacy risks with high potential of harming the users' data and services, where the infrastructure offered by an untrusted provider may be a hostile environment, where in security requirements cannot be ensured [100]. Hence, despite deploying appropriate security mechanisms, extra measures should be provided to gain users' trust regarding all involved services and providers.

4.4. Threat analysis

Though MC has great potential to offer businesses economical and scalable IT solutions, it presents various security risks. This section analyzes the potential threats leading to security breaches by outsiders or any of the entities in the model. Threats against interactions in this model can be classified into four categories: threats related to standards and regulatory authority, threats related to providers, threats related to access, threats related to Data and Network-related threats [47].

● Currently, MC lacks proper security standards or suffers compliance risks in cases where security standards are defined due to a lack of governance for audits [47], [37], [19], [71], [18].

● CSP related threats are related to the services offered by a provider and the ways they can be exploited. These threats include insecure APIs which facilitate users' access to many offered services [47]. This may lead to unauthorized access to services by malicious users or the unlawful release of users' data. Multi-tenancy nature of CC and MC means that Software and Hardware

resources are shared between various users. Malicious tenants may use any vulnerabilities in the virtualization layer of one or more provider infrastructures to gain unlawful access to other users' services or data.

- Data related threats may include data redundancy, data leakage, and data unavailability.

- Network related threats have a great impact on the security of both the CC and MC model due to their high dependency on networking and Internet communications. Service hijacking may result from vulnerabilities in communication protocols leading to leakage of data or loss of service to legitimate users.

- Access related issues include unauthorized access (impersonation) by a dishonest administrator or another user within the same provider. For MC, unauthorized access may occur by other provider users.

Accordingly, the security threats to MC model can be enumerated as follows:

a) **Impersonation of a user:** A malicious outsider may steal the identity of a legitimate user and try to login into the management console on his behalf. This can be accomplished through a phishing or pharming attack, or PWD guessing. It may lead to leakage of the user's data, loss of service to the user's beneficiaries, or hijacking of resources allocated to the user. Another type of impersonation may occur during $\boldsymbol{REQ_U}$ messages to access services paid for, allowing a malicious outsider to hijack the service.

b) **A malicious insider impersonation**: A malicious insider may be a dishonest administrator within the CSP, or it may be another legitimate user or admin of the provider. A malicious insider may hijack a service or resource allocated to user $\mathbf{u_1}$ by gaining unauthorized access to them. Additionally, this may occur due to vulnerabilities in the virtualization layer of the provider infrastructure, where a scheduler favors $\mathbf{u_2}$ requests over $\mathbf{u_1}$ requests. Also, lack of secure encapsulation may lead to granting $\mathbf{u_2}$ unauthorized access to $\mathbf{u_1}$'s resources or data.

c) **Impersonation of a CSP:** An attacker may impersonate a legitimate **CSP** during communication between a user **u** and **CSP**. While a *RES* is being sent to the user in response to an access request, an attacker may hijack the communication channel to gain unlawful access to **u's** assets or lead to loss of service to **u**'s beneficiaries.

d) **Altering *REQ* or *RES* communication messages:** An alteration may include replaying, delaying, or modification of messages in an attempt to gain unlawful access to, or misuse **u's** privately owned or cloud hosted assets. Assets refer to data as well as other resources such as servers or bandwidth. This may occur due to vulnerabilities in communication channels and protocols.

e) **Eavesdropping:** Eavesdropping on communication channels by an attacker to get private information of **u** or **CSP** to create a later attack. This could also occur during CSP-to-CSP communication, leading to leakage of user's data or loss of services. *REQ* or *RES* communication messages may contain vital data such as secret security credentials, financial information, or vital data about the user.

f) **Unauthorized access to data at rest:** A malicious insider or outsider may unlawfully gain access to data stored on the provider's storage resources. This may lead to the leakage of private or sensitive data, causing financial loss to both users and CSPs.

g) **DoS attacks:** An outside attacker or a different legitimate user $\mathbf{u}_2$ of the **CSP** may hog resources or communication channels used by a legitimate user $\mathbf{u}_1$. Due to the high dependency of MC on HTTP communication and REST architecture, the attacker can flood the communication channels and web servers with HTTP requests [47].

h) **Disputes:** Intentionally or erroneously false requests made by a legitimate user may lead to financial loss or may affect another user of the CSP. On the other hand, intentionally or erroneously false requests made by the **CSP** may cause financial loss or the denial of legally paid resources allocated for the user. Due to the lack of proper security standards and lack of governance for audits, the fear of unresolved disputes affects the decision of new users to start taking advantage of the services and benefits offered by the MC paradigm.

4.5. Observations

Based on the generic model described in Section 4.3 and the threat analysis in Section 4.4, we can make the following observations:

Observation 1. The model presents multiple heterogeneous entities with varying IT capabilities, number of users, and business models. This is because MC users may belong to a hospital, a university, or a multinational company, to name a few, each of which may have a few users or

hundreds of users. It may have a privately hosted data centre or just rely on PCs provided to its employees.

Observation 2. The interacting entities communicate over the Internet. They have no standardized, unified way of communicating within the cloud context. Hence, they may utilize all and any protocols standardized for web communication and web services. They are also vulnerable to all risks and threats applicable to Internet communication.

Observation 3. The types and number of security credentials necessary to secure the interactions among the model entities are too diverse and doesn't fit the model variable nature. As per the description of the credential structure in Section 4.3.3, there are six credentials that may be utilized. Credential types vary between PWDs, OTPs, uncertified PUK/PRK pairs, X.509 PUK and matching PRK, TAKs, and multiple shared SAKs.

Observation 4. This security scheme is complicated with long processes and many keys to create and manage (observation 3). Hence, this complicated scheme may be difficult to apply for low-resource users, such as a small business or a mobile user.

Observation 5. There are two phases of authentication in a user-to-CSP interaction. The first phase is initial authentication to gain access to the management and security console, which is done during login to U-S as described in Section 4.3.5 (login). The second is to access services agreed upon after successful login, where all requests from the user are authenticated to the CSP. This kind of authentication is ongoing for as long as a user is utilizing a service. Examples of ongoing authentication are presented in section 4.3.5 (requesting resources, user accessing services, services accessing services)

Observation 6. The ongoing authentication of service requests is one-way since it only authenticates the user to the service.

Observation 7. Collaboration between various CSPs doesn't currently follow processes or protocols standardized for CSP-to-CSP communication, which may jeopardize the data and processes of the user that are necessary to finish the task at hand [60].

Observation 8. Due to competition, the information presented by most current CSPs doesn't cover the detailed methods and processes followed during service provision to the user [60]. CSPs

don’t provide sufficient information on processes that may be employed for P-to-P collaboration [5].

4.6. Security Requirements

Examining the threat analysis and the observations made on the Generic Model of MC presented in Sections 4.4 and 4.5 respectively, we can derive a set of security measures that can be taken to mitigate those risks. Of course, an effective security solution should not create new security concerns nor incur high-performance costs. Since authentication is the goal of this research, we will present a more in-depth analysis of authentication-related requirements in Section 4.6.1. Section 4.6.2 emphasizes the other requirements necessary for a complete security solution for MC.

4.6.1. Security requirements related to Authentication

This section will derive a set of desirable security requirements for designing an effective authentication solution for MC.

R1: Mutual Entity Authentication:

The purpose of entity authentication is to ensure that every interacting entity has the identity it claims. This should be ensured for all entities’ interactions in a mutual manner:

- User to management console (u-to-M-CSP)
- User to service (u-to-S-CSP)
- Service of one CSP to a service of another CSP (S_a-CSP_x- to-S_b–CSP_y)
- Service to another service of the same CSP (S_a-CSP_x- to-S_d–CSP_x)

R2: Continuous Authentication Process:

To guarantee that all cloud resources allocated to a user are protected against unauthorized or malicious access, the authentication process should continue throughout the resource utilization operation. The authentication process should include:

- Registration (Identification): This step takes place first to create credentials and exchange security information between two communicating entities.

- Initial authentication: This should be made at the beginning of communication between two parties, such as a user logging into a **CSP** service, a user requesting access to service **S**, or a **CSP_x** accessing a service provided by another **CSP_y**. The two communicating parties should use the credentials and information exchanged during the registration phase.
- Ongoing authentication: As two entities continue the interaction to conclude a task, the authenticity and validity of all requests and responses should be ensured. This applies to all entities, whether they are a part of the same domain or cross-domain entities.

R3: Non-Repudiation:

It is a security service that protects against false denial of taking part in communication by one of the interacting entities. Non-repudiation of origin ensures that the sender doesn't deny sending a message. Non-repudiation of receipt ensures that the receiver of a message doesn't deny receiving the message. Hence, both interacting parties should have proof that the other entity took part in the interaction. Both types of non-repudiation must be ensured for all communications between interacting entities:

- User to management console (u-to-M-CSP)
- User to service (u-to-S-CSP)
- Service of one CSP to a service of another CSP (S_a-CSP_x- to-S_b–CSP_y)
- Service to another service of the same CSP (S_a-CSP_x- to-S_d–CSP_x)

R4: Interoperability:

Interoperability refers to the ability of heterogeneous systems to interact and collaborate efficiently and securely. For M2C, it would allow users to employ the services offered by multiple CSPs in a seamless manner. Hence, an authentication solution should ensure secure cross-domain communication. Interoperability is achieved through well-documented and well-tested specifications.

R5: Performance:

Any security solution should not hinder service provision; rather it should enhance it. It should consider low-resource users, such as those on mobile devices. Users accessing services from a mobile device should have the same experience as non-mobile users [1]. In addition, ongoing

authentication should be as seamless as possible, requiring the least amount of user intervention. Thus, the solution should be low weight in terms of:

- The computational cost for the user: by reducing the computational procedures.
- Communication messages overhead: by reducing the number of exchanged *REQ* and *RES* messages as well as the size of the messages.

4.6.2. Other Security Requirements

This section will provide more requirements necessary for a complete security solution for MC, to counteract the threats specified in the previous threat analysis. These requirements are:

R6: Privacy:

Privacy refers to the right of an individual or organization to keep information of a personal or proprietary nature from being disclosed to others. Privacy may be at risk in MC due to the interconnected nature of MC and the multi-tenancy nature of CC in general. Users may be worried about the leakage of personal information collected through advertisement pushes [55]. Encryption and isolated storage of user's data may be used to protect their privacy [94]. Another privacy concern is the illegal disclosure of location information by applications [55]. Spatial cloaking solutions are suggested by many researchers to guard against this problem [55], [24]. An efficient MC security solution should be able to ensure the privacy of users and user organizations while allowing the operation of user's data to provide the requested services.

R7: Confidentiality:

This security requirement should apply to all data in transit or at rest. It should ensure the secrecy of the data and protect the privacy of users and user organizations. Confidentiality should apply to all data stored by the CSP, all communication messages between users and the CSP, and all messages from one CSP to another CSP. Data Confidentiality is usually provided through cryptographic techniques, including Public Key encryption, Secret Key encryption, hash function, or a combination of those techniques [83].

R8: Availability:

This security requirement is to guarantee that a cloud service S or a cloud resource is accessible and usable when an authorized entity calls for its use. The context of MC poses an extra risk of

losing the availability of services due to the elasticity of CC with a large number of users trying to access services. Mobility of the user over networks with varying parameters and security measures introduces more risk as well. Availability should be assured for all services offered by a **CSP** including the **CSP** website for registration or logging into the management console. Measures should be taken to provide availability and protect against DoS attacks.

4.7. What is Missing?

Based on the analysis of authentication methods for distributed systems presented in the previous chapter, we will evaluate the distributed systems authentication systems presented in Section 3.5 against the security requirements for M2C, identified in Section 4.6. Accordingly, we make the following observations:

- All authentication methods for P2P presented in Section 3.5.1 do not satisfy most of the requirements for M2C. This is because P2P is based on the idea of anonymous nodes with equal weight (peers) collaborating for a common goal. Hence, the main goal of P2P authentication is fast operation, achieved through high connectivity within peer nodes. Therefore, authentication is required for different reasons and at different security levels than that required for M2C.
- Although M2C features are very similar to those of GC, out of the authentication methods for GC presented in Section 3.5.2, Kerberos is the scheme that covers most requirements. It doesn't provide continuous authentication or non-repudiation. It can be augmented with other techniques to be a better fit for M2C requirements.
- As for the CC methods presented in Section 3.5.3, they are based on the same concepts as those in GC. Based on the examination of the presented methods, it is apparent that simple single-factor methods don't meet the complex requirements of M2C. The most promising schemes may be inter-cloud methods and lightweight ID-based methods.
- It should be noted that requirements R2 (message authentication) and R3 (continuous authentication) are not met by any of the methods. R2 can be established through the application of MACs or Hash functions to the entire message rather than just the keys. R3 can be accomplished with a simple modification to some of the discussed methods.

Table 4-9 shows the authentication methods reviewed (in rows) and authentication requirements R1 to R5 (in columns).

Table 4-9: Current Research Authentication Methods Evaluated against Requirements

		R1	R2	R3	R4	R5
P2P	Enc PWD [11]	×	×	×	×	✓
	PKI based [44], [54]	✓	×	×	✓	×
	IBC [54], [65]	✓	×	×	×	×
	CL-PKC [54]	✓	×	×	✓	×
GC	Static PWD [29], [39]	×	×	×	×	×
	Dynamic PWD [29], [39]	×	×	×	✓	×
	Kerberos [83], [29], [52]	✓	×	×	×	✓
	PKI based [29], [39], [14]	✓	×	×	✓	×
	SPTC [56]	✓	×	×	✓	×
	IBC [29], [41], [35]	✓	×	×	×	×
	CL-PKC [29], [30]	✓	×	×	✓	×
	Pairing-free CL-PCK [30]	✓	×	×	✓	×
CC	Multi-level PWD [26]	×	×	×	×	×
	TCG [81]	×	×	×	✓	×
	CAM [13]	✓	×	×	×	×
	Dynamic Multi-factor [10]	×	×	×	✓	×
	InterCloud [17], [57]	×	×	×	×	✓
	ECC ID-based [22]	✓	×	×	×	✓

Table 4-9 confirms a gap in the authentication provision for M2C as per the requirements. All methods examined don't meet all requirements related to authentication R1 through R5. Most methods fail to provide continuous authentication (R2), and non-repudiation (R3) measures. Non-repudiation can be provided by means of digital signatures. Interoperability (R5) is a key requirement that is not achieved by most examined methods.

Hence, we need to propose a comprehensive authentication method for the context of M2C. The method needs to be fine-grained for this computing environment, considering the limitations of the context and harnessing the power of its unique features.

4.8. The Best Way Forward

Looking back at the threat analysis presented in Section 4.4, many of the security threats identified in the threat analysis can be remedied through a strong and effective authentication solution. All impersonation attacks mentioned in threats A, B, and C can be counteracted if effective authentication is employed. Impersonation attacks, if successful, can lead to more attacks on the resources of the Client. Moreover, message alteration (threat D) may not be prevented by strong authentication but may be detected to save interacting parties from any harmful consequences. Also, as threat F states, unauthorized access to data at rest can be stopped with strong authentication. Furthermore, (threat H) can also be resolved with the aid of an effective authentication solution.

Analysis of current research on authentication solutions for distributed systems in Section 4.7 helps us identify the gap in the current knowledge, for meeting M2C authentication requirements. Furthermore, it paves the way for setting design requirements for an effective authentication solution that takes advantage of the best features of state-of-the-art authentication methods and bridges the gap. Hence, the best way forward is to provide a sound, reliable elastic authentication solution. The solution should use the special features of the M2C context to our advantage. Security requirements should be met without introducing any new risks.

4.9. High-Level Ideas

Based on the threat analysis of the MC Model presented in Section 4.4 and the security requirements derived in Section 4.6, we propose a novel architecture for authentication solution for MC. The goal of an authentication system **(AS)** is to decide whether the identity of an entity ***e***

trying to interact with another is authentic, i.e., they are who they present themselves to be. The authentication decision is made based on the evidence offered by ***e*** and data collected by **AS** to trust ***e.*** For MC, entities interacting are distributed among several providers, each with a different set of services and security mechanisms, and objectives [100].

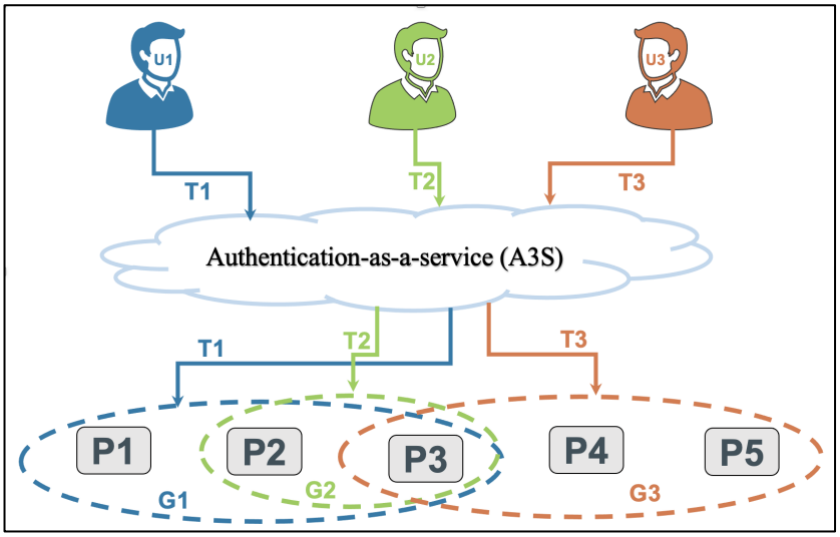

Figure 4-12. A logical view of A3S cloud

For the MC authentication solution, we propose to provide cloud-based authentication, or authentication-as-a-service cloud (A3S). This allows for a shift from the usual provider-centric cloud paradigm into a more balanced approach focused on all interacting entities users and providers alike [51]. As Figure 4-12 depicts, A3S would represent "a security distribution layer" for authentication provision, providing an end-to-end interface for the multi-cloud between user-centric and provider-centric views [51]. So, based on task **T** initiated by user **u**, A3S facilitates authentication services for all interactions necessary between **u** and all CSPs in a cloud provider group **G** deployed to accomplish **T**.

The proposed A3S architecture fulfils the distribution promise, as with all cloud services. It is also dynamic, as it offers authentication on-demand as required within the current dynamic context of the communicating entities, including users and providers. A third aspect of the proposed solution is elasticity. Elasticity in cloud computing refers to the level of adaptability of computing solutions in response to changes in demand metrics by provisioning/de-provisioning resources seamlessly [36]. However, our work adds other dimensions to the previous definition of elasticity by allowing various levels of security provision, for authentication specifically, and performance variations to

meet users' demands and needs. Security elasticity can be achieved by allowing for different levels of assurance, multiple methods of authentication with various types of credentials, and a variable number of trusted entities. Performance elasticity is accomplished by allowing for a varying number of interacting entities, including users, services, and providers. The computational cost, number, and size of authentication messages can vary in accordance with entities' requirements and limitations. Hence, it is an Elastic, Dynamic and Distributed authentication solution. This solution should be invoked for a given interaction within MC to authenticate the communicating entities as per their required assurance level and use the available authentication methods and tools.

The next chapters will introduce the novel framework TruE for establishing trust for interacting entities in M2C based on three aspects: credentials such as -passwords and cryptographic techniques- as well as an entity's reputation based on previous interactions and recommendations of other entities within a group **G.** Reputation and recommendation are measured and collected after any authentication instance and saved in the Authentication service Database. The trust framework TruE is the main component of a novel architecture (END AuthN) Cloud service, which identifies credential based LoA-related factors and defines relationships between them. It includes a novel methodology for structuring these factors in order to combine them for the purpose of computing an aggregate value of trust based on these factors. END AuthN makes authentication decisions for requesting entities by comparing aggregate trust value with the trust requirement of a given entity.

4.10. Chapter Summary

This chapter discusses in depth the identification of the research problem. We presented and analyzed a real-life use case of M2C to understand the context and security provisions within M2C. This allowed us to describe a generic model for M2C, the MC-model. Next, we conducted a threat analysis of MC to arrive at a set of security requirements for MC, with a dedicated section for authentication related requirements. This allowed for an analytical comparison of authentication solutions presented in section 3.5 and thus assist in identifying the knowledge gap in authentication solutions for M2C and deduce the best way forward in Section 4.7. Finally, we presented high-level ideas for our novel solution to authentication provision for the MC-model in Section 4.8.

The next chapter provides a review of Trust and details the design of a novel framework to establish Trust for M2C.

Chapter 5: Trust as basis for authentication: Review and Design

5.1. Chapter Introduction

Trust plays a major role in securing resources and transactions in M2C. Trust management needs to be an essential part of any M2C security system, especially Authentication. Authentication is often based on establishing trust relationships between communicating entities. This chapter defines trust and reviews trust models that may be employed in distributed systems such as P2P, Grid, and Cloud Computing. Section 5.2 presents a Definition of trust followed by Section 5.3, which describes how trust is established. Section 5.4 provides a review of various trust models for distributed systems. Section 5.5 presents discussion and observations for these models. Section 5.6 presents the concept of three modes of authentication. Sections 5.7 through 5.10 cover the design elements and description of our novel authentication solution based on trust.

5.2. What is Trust?

Definition of trust varies according to the context in which it is used. Parallel to what it means in a social and psychological context, trust in Computer Science refers to confidence in the integrity and surety of a person or an entity. Hence, in the milieu of distributed systems such as M2C, trust can be defined as the level of confidence placed on an entity **e** participating in transactions with other entities within the system [3]. This level of confidence is then used by other entities to make decisions such as:

- whether to accept transactions with the entity in question **e**,
- what type of transactions it holds with **e**,
- and what type of resources it can share with **e**.

For CC generally and M2C particularly, trust is integral since inter-entity transactions may span various organizations and domains, some of which may or may not be trusted to the same level [79]. Moreover, trust should be considered for clients requesting or utilizing services in order to

protect CSP and other clients' resources. For the same reason, trust should also be established for resources or services allocated to perform a given task for a client.

5.3. Trust Management

Due to the importance of trust, trust management is one of the most challenging issues in Cloud computing. Trust management includes the following essential tasks [63], [27], [3]:

- Collect necessary information about entities within a system,
- Identifying factors used to quantify trust,
- Establishing trust for those entities using identified factors,
- And Dynamic supervision of an existing trust relationship.

Establishing trust depends on how it is perceived. Trust may be perceived in various ways [90]:

• Trust as a risk factor, where it is defined as a prediction of an entity's future actions.

• Trust as belief, where it is the willingness of an entity to act in accordance with other entities' actions.

• Trust as subjective probability, when it is conveyed as a specific level of likelihood that an entity will perform a given task within a certain context.

• Trust as a transitivity relationship, when it is a weighted binary relation between two entities in a network.

When Trust is established, it is quantified and expressed as a value. These measured outcomes are known as trust metrics and can be articulated in numerous methods [3], [27], [90], [38]:

- **Trust scale:** The level of trust is measured and expressed as a continuous or discrete value. For discrete value metrics, trust level is expressed as *v*, where $v \in$ a discrete set of values such as $\{highly\ trusted, meduim\ trust, not\ trusted\}$ or $\{0,1\}$. For continuous metrics, trust level is expressed as *v* where *v* falls within an interval *[a,b]* where $a \neq b\ and\ a, b\ \in \mathbb{R}$. Threshold based scales express the value of trust as acceptable only when it approaches a predefined threshold; so, an entity is trusted if its trust value $v \geq T\ where\ T\ is\ trust\ predefined\ value.$
- **Trust facet:** A trust facet may denote the trustworthiness of an entity e as the shortest distance from origin to *(T, C)* on a two-dimensional rectangular plane, where *C* is a confidence value in the interval *[0,1]* and *T* is a trust value in the interval *[0,1]*. Another method is to denote trust value

as a triplet (b, d, u); b, d, u fall within the interval *[0, 1]* and b+d+u = 1, where b, d, and u represent belief, disbelief, and uncertainty, respectively.

- **Trust logics:** Many approaches use probability or fuzzy logic as a metric for trust. In one study, the trust metric for ad-hoc networks was defined as the ratio between the number of packets forwarded correctly and the total number of packets received [95].

Accordingly, trust models are defined by identifying how each entity **e** within the system perceives and establishes trust and the metrics used to express the value of that trust.

5.4. Classification of Trust Models

Trust models are defined based on how trust relationships are established between entities in the system. However, methods for establishing trust may be classified by different criteria.

One way of categorizing trust is based on the objectivity of the techniques used to establish trust leading to two classes: hard trust or soft trust. Hard trust is calculated based on objective hard techniques like security algorithms, audits, and certificates, while soft trust is more subjective since it is based on behavior like user feedback and reviews [77].

More commonly, trust between entities usually falls into one of two categories: direct and indirect approaches. In the direct approach, two entities build credibility based on the history of interactions they shared. Accordingly, the ensuing value of trust is also known as Native trust or Direct trust [25]. Conversely, under the indirect approach, third parties predict user needs or interests and make recommendations to aid users in service selection and composition [85]. These recommendations are made based on various criteria such as service ratings provided by other users with similar needs to those of the given user. And in the cases where indirect Trust is necessary, we need to consider the Transitivity of trust, where direct trust may be propagated from one entity to another. Propagation of trust has two basic operations [105]:

- **Trust Concatenation**: If E is the evaluator entity and Z is the evaluation target, which is not directly known by E, then an intermediate entity M, which knows Z and is known to E can provide trust information. Taking advantage of transitivity, a significant increase in the coverage of trust relations among entities within a system is observed. Figure 5-1. shows the concept of Trust Concatenation.

Figure 5-1. Trust Concatenation

- **Trust Aggregation:** Trust aggregation is the sum of trust propagated from multiple trust relations. This way, two or more entities can pledge for another entity Z under evaluation by the evaluator E. Figure 5-2. shows a simple graph representing the concept of trust aggregation, where two entities, X and Y can vow for entity Z under evaluation by E.

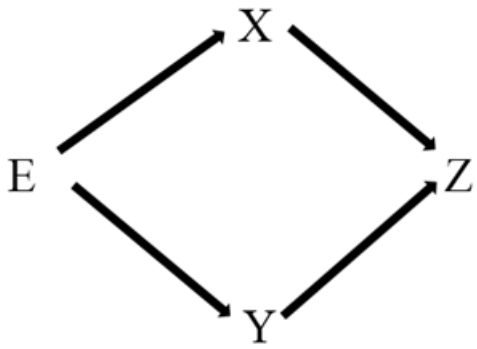

Figure 5-2. Trust Aggregation

The following subsections will review several trust models under the direct and indirect approaches for distributed systems such as P2P, Grid, and Cloud Computing. They will provide a review of varied trust models, where direct and indirect approaches are combined to establish trust for entities in a system.

5.4.1. Overview of Direct Trust Models

This section will present trust models where interacting entities build trustworthiness based on the interaction histories, they share in a direct manner without the need for the input of any third parties.

1- Trust and Reputation model for Agent-based Virtual OrganizationS (TRAVOS) is a probabilistic based model [87]. In TRAVOS, Trust is defined as the probability of successful interaction between two agents. Past direct personal experience is used to calculate the trust between two agents if no direct history is recorded and witness evaluations are used as a last resort

[66]. To account for the possibility of inaccurate evaluations, an evaluator agent compares its own observations with those of witnesses to be able to assess the level of reliability of the witness's beliefs. The assessment is then used to calculate the weighted impact of a witness's beliefs about an evaluated agent. However, TRAVOS assumes that the behavior of agents does not change over time, which means that trust evaluations become less reliable with time [66]. Further, it also doesn't point out any method to solicit witness evaluations and assumes that evaluations are accessible as needed [66].

2- In Geetha and Jayakumar model, a host evaluates other hosts based on direct experience [31]. This model follows a decentralized approach. In the model, a trust value of 1 is assigned to a trusted host, while a value of -1 is assigned to a distrusted host. To keep track of evaluations, each host has a routing table containing a list of all other hosts evaluations as well as its own. The overall trust is the sum of the evaluators' opinion in addition to the evaluations of other hosts in the network. Feedback from malicious hosts can be tolerated in this scheme as long as the number of malicious hosts is less than half the total number of hosts in the network. One of the pitfalls of the system is the overhead of exchanging information about every evaluation with every other host in the system, even those who might not need it. Moreover, all hosts need to be informed of hosts who are added or removed from the network to update the routing tables. Another disadvantage is that it gives the same weight to all evaluations, regardless of how recent they are.

3- The Local Trust Model, which is based on the time influence function, in P2P Networks bases trust evaluations on the Eigen Trust model [25]. The improvements proposed are to reduce network overhead, enhance the binding force of malicious entities, and make the model more reliable and accurate.

Eigen Trust for a network of *n* nodes uses the following method to calculate the local trust value denoted by s'_{ij} [25]:

$s'_{ij} = A_{ij} - B_{ij}$, where A_{ij}, B_{ij} are the accumulated number of satisfied evaluations and unsatisfied evaluations of node i to node j respectively.

$$s'_{ij} = w \times \frac{s_{ij}}{\sum_{j=1}^{n} s_{ij}} + (1 - w) \times \frac{1}{n}$$

This will account for malicious entities providing dishonest evaluations. However, it doesn't differentiate between malicious and non-malicious false evaluations. Further, it doesn't take into

account nodes with no prior history, such as new ones. To resolve this, Cui and others suggested the addition of a trading frequency and time influence function [25]. These functions add to the frequency of interactions between entities over a given period of time, since more interactions implies more credible and reliable evaluation.

5.4.2. Overview of Indirect Trust Models

As opposed to trust models presented in the previous section, this section describes trust models where interacting entities build trustworthiness based on the interaction's history, they collect indirectly from third parties. These models are usually referred to as recommendation-based models.

1- A recommendation based model is suggested for Mobile Ad hoc Networks MANETs to limit the effect of dishonest entities on the performance of the trust model [78]. The model proposes a defense scheme to filter out dishonest recommendations like bad-mouthing and ballot-stuffing. Bad-mouthing refers to the action of a group of nodes conspiring to propagate negative ratings of good nodes to tarnish their reputation in the network. On the other hand, ballot-stuffing refers to the propagation of unfair positive ratings for poorly performing nodes in the network. To accomplish this, recommendations are accumulated over a period of time to ensure consistency. The proposed model uses a Bayesian statistical approach for computing trust values, assuming they follow a beta probability distribution. The beta distribution is estimated using two parameters: α, β, where α represents the accumulation of positive observations (forwarded packets) and β represents the accumulation of negative observations (dropped packets). They define the distribution by the gamma function:

$$f(p|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)(\beta)}\, p^{\alpha-1}\,(1-p)^{\beta-1}$$

The proposed model clusters recommendations are based on three different criteria: the number of interactions using the confidence value, compatibility of information with the evaluated node using the deviation test, and closeness between entities [78]. The use of various criteria to judge the dishonesty of a node aids in mitigating the influence of false negative and false positive ratings. The Confidence value starts at 0 when there are no observations between entities and gradually increases as the number of recorded observations upsurges [78], [88]. The Deviation value

represents the compatibility between the received recommendation and the personal experience of the evaluating node [78], [15].

5.4.3. Overview of Varied Trust Models

Varied trust models refer to models where direct and indirect approaches are employed to establish trust for entities in a given system.

1- FIRE [40] is a trust reputation model used to assess agent performance in multi-agent systems according to four values: Interaction Trust (IT), Roles based Trust (RT), Witness Trust (WT), and Certified Trust (CT). IT is a direct metric calculated from previous evaluations of direct interactions, with each value assigned a particular weight. The weight of an evaluation depends on its freshness where, recent evaluations receive higher weights. RT is calculated based on environmental rules set by the system's designer to improve performance. Hence, the weights of RT values are static and set by the designer. The values represent the certainty of the rules. WT are evaluations of directly related agents, referred to as witnesses. CT evaluations are similar to WT evaluations, but they come from certified Trusted agents rather than random witnesses in the system. The overall trust is the weighted mean of IT, RT, WT, and CT, where the weights are based on trust reliability and coefficients defined by users to denote the importance of each value. On one hand, the FIRE model improves the performance of agent systems [80], whereas on the other hand it does not provide details about the agent evaluation process nor assigns the weights of the four trust values. Also, the model is optimistic and doesn't account for false evaluations. Moreover, FIRE is a static model as it relies on many static parameters, disregarding the dynamic nature of the environment it should operate in [66].

2- Jurca and Faltings Trust model relies on direct experience and reputation [45]. This model relies on centralized agents defined in the network to gather reputation reports from agents. A payment method is proposed to encourage honest evaluations by increasing or decreasing monetary rewards based on honest evaluations. This money is used by agents to buy reputation reports from centralized agents. When untrustworthy agents provide dishonest evaluations, they lose their money which prevents them from further purchasing the reputation reports. This scheme does encourage honest agents to aim for accuracy, but it doesn't account for malicious agents, who wouldn't care to lose the money, as long as they disrupt the system's behavior [66]. Another

disadvantage is that this model gives equal weight to all evaluations as reports are aggregated through averaging.

3- Regret [76] is a decentralized Trust reputation model that calculates the trust of an agent based on direct experience and indirect reputation measures of trust. This model uses a social structure called a sociogram that represents social relationships such as cooperation, competition, and trade in a graph where the entities represent the participants and the edges denote the nature of relationships among them [66]. Indirect reputation evaluations are based on witness reputation, neighbourhood reputation, and system reputation. The witness reputation refers to other agents' beliefs. The neighbourhood reputation is based on fuzzy rules' evaluations of neighbouring agents which have previous interactions with the evaluated agent. The system reputation is used to find the default reputation of an agent based on its behavior or the role it plays in a social group, in a manner similar to the concept of RT in FIRE [80]. Regret calculates the reliability of trust values to improve the decision-making process. It also considers the case of untruthful witnesses providing false information by evaluating their credibility. A disadvantage of this model is that it assumes that agents' behaviour doesn't change with time. It also uses static weights to combine the four evaluation values, which may cause failure when agents act unexpectedly or when the environment changes.

4- A trust management model was proposed for Cloud Computing as Service Oriented Architecture SOA [85]. This is a hybrid model that aims to ensure entities' credibility before transactions. It also evaluates trust values based on predefined measures or measures requested by Cloud users. They also consider both Cloud users and Cloud providers perspectives in trust calculation and integrate both hard and soft trust through boxing evaluation techniques. A caching technique is used to optimize querying services. The proposed model includes several novel techniques, including the sliding window technique, the Service Level Agreement (SLA) trust calculator, the prediction technique, and the boxing evaluation technique. To implement the sliding window technique, only valid interactions (negative or positive) are used to calculate trust, with the negative interaction window being bigger than positive interaction window. The SLA trust calculator processes feedback from SLA manager to generate a common trust value to be stored in a trust repository. It uses direct trust for each defined SLA between the Cloud provider and Cloud users as per defined key performance indicators (KPIs). KPI scores are normalized to be from 0 to 1 and assigned weights by Cloud users. The prediction technique is used when there are

no saved trust values for a certain service to find equivalent properties in peer services. And finally, the boxing technique is used to calculate trust value [85], [34].

5- The Trust Model proposed by Basheer et al. evaluates the level of confidence of an agent based on local confidence (LC), based on direct trust, and global confidence (GC) values, based on indirect trust [12]. Both LC and GC are calculated based on three values: the importance of the value (I), Trust (T) and certainty (C). T and C are calculated as percentages of the satisfaction of predefined rules called factors and evidence. The overall confidence level is the sum of the LC and the GC. This model is decentralized. It also assigns static weights for LC and GC. The case of malicious agents or false evaluations of witnesses is overlooked in this proposal [80].

6- A proposed Trust Model was implemented on CloudSim and based on Service Parametric Model [79]. This model integrates the role of resource brokers and computes Trust values for different Cloud Service Providers CSPs. The Cloud user provides a list of parameters based on the capability of CSPs, the behavior of CSPs and feedback from users with each submitted job. Capability parameters include the speed, efficiency, and power of CSPs. Behavior parameters are Availability, Reliability, Success Rate, and Throughput. Feedback is based on ratings given by users and brokers. Each parameter is assigned a weight W_c, W_b, and W_f respectively, where

$$W_c + W_b + W_f = 1 \text{ and } W_c, W_b, W_f < 0.5.$$

A Cloud coordinator component then collects the parameters and weights, parameters, and the incoming jobs and divides the jobs into N different equal-size cloudlets. The Cloud coordinator forwards these weights, parameters, and cloudlets to a trust estimator component. The user/broker provides the values for the parameters. The trust estimator calculates all three types of trust values, capability-based T_c, behavior-based T_b, and feedback-based T_f, and averages of trust values for each CSP and then returns these values to the broker. The broker selects the CSP that has the maximum average trust value and returns that CSP to the trust estimator. The trust estimator calculates the aggregate trust value T of the selected CSP. The trust estimator then finally returns T to the user through the Cloud coordinator and updates the T value in the database. T is calculated as follows:

$$T = Wc*Tc + Wb*Tb + Wf*Tf.$$

5.5. Discussion and observations

Sections 5.3.1, 5.3.2, and 5.3.3 presented a review of various trust establishment models. Based on the types of interactions used for trust calculations, trust models were classified. Some models rely only on direct interactions between evaluating and evaluated entities, while others need more entities' feedback to make decisions. This feature is significant as it may add flexibility or limitations to the system [80]. trust models can also be categorized on the basis of weight assignment, handling false evaluations, or centralization [80], [34]:

- **Weight assignment**: Some models assign weight to various trust evaluations dynamically, while others rely on static weights. Assigning weights dynamically adds flexibility and helps avoid making wrong decisions that may affect the system's functionality.
- **False evaluations**: Some models assume all entities are honest and provide correct evaluations. False evaluations negatively affect the system's functionality and security.
- **Centralization:** Trust may be calculated in a centralized or decentralized manner. The decentralized approach adds flexibility to the system and evades the single point of failure issue that may threaten the functionality of centralized systems.
- **Trust customization** [34]: Trust is customized according to the source of evaluation provided. Trust values can be local and subjective to the entities providing them, or global and independent of the evaluator of trust.

Table 5-1 shows the trust establishment models examined (in rows) and relevant characteristics (in columns).

Table 5-1: Summary of Trust model reviewed

Classification	Model	Environment	Centralized	Consideration of Direct approach	Consideration of Indirect approach	Consideration of LoA	Consideration of False Evaluations	Weighted evaluations
Direct	TRAVOS [87]	Agent	×	✓	×	×	✓	✓
	Geetha and Jayakumar [31]	Agent	×	✓	×	×	Tolerated if malice	×

							hosts < half hosts	
	Local Trust Model [25]	P2P	×	✓	×	×	✓	×
Indirect	A recommendation based model for MANETs [78], [88]	MANETs	×	×	✓	×	✓	×
Hybrid	A trust management model as SOA [85]	CC	×	✓	✓	×	_	×
	FIRE [40]	Agent	×	✓	✓	×	×	×
	Jurca and Faltings [45]	Agent	×	✓	✓	×	×	✓
	Regret [75]	Agent	×	✓	✓	×	✓	×
	Basheer et al. [12]	Agent	×	✓	✓	×	×	×
	Model based on Service Parametric Model [79]	CC	✓	✓	✓	×	_	✓
	Yao & Zhang [103]	Grid	✓	×	×	✓	_	✓

Examination of Table 5-1, we make the following observations:

O1. Most trust Models for CC concentrate on the Quality of Service or QoS metric.

O2. Most trust Models for CC that have been reviewed aim to examine trust Levels for Cloud Services and Cloud Service Providers to aid potential users in choosing the most trusted services offered.

O3. Recommendation based trust techniques aid in discovering misbehaving entities before interaction, so that the potential bad experience can be avoided. Another advantage is that recommendations provide more information about the evaluated entity, giving more confidence in the decision to interact with the evaluated entity [78].

O4. Most trust models don't consider credentials when quantifying trust for an entity in question.

5.6. Three modes of Authentication

Based on the on our analysis and observations of the reviewed trust models, we identify three modes of authentication: credential-based, reputation-based, and recommendation-based

authentication. For credential-based mode, any factor relating to credentials presented by a claimant for the purpose of identity verification will affect LoA, thus influencing the trust value established for an entity **e**. Reputation-based mode is a direct approach, where the authentication decision for an entity **e** requesting access to cloud service **S** is guided by the previous history of interactions between **e** and **S**. Factors relating to the reputation-based mode will also influence the trust value established for an entity **e**. On the other hand, recommendation-based authentication mode is an indirect approach, where third parties rely on previous interactions with the entity in question **e** to make a recommendation on whether to take part in an interaction with **e** [42]. All such recommendations will influence the value of trust established for the entity in question. These recommendations are usually based on service ratings provided by other clients with similar needs to aid a new or potential client in choosing the best service. Our research extends the scope of recommendations to include services recommending a user and attesting to their fair dealings. All research reviewed bases trust on one or two of these above discussed modes. However, we propose a novel authentication solution based on trust accumulated from all trust factors available at the time of the authentication request. So, all factors collectively will help decide how much trust we can place in entity e. If the level of trust matches the required trust value for the transaction in question, then a positive authentication result is achieved. This fulfills the promise of an elastic authentication solution based on the information available in the context of a given authentication request.

5.7. Design Requirements for an Elastic and Distributed Authentication Architecture (END AuthN)

Based on our analysis of the use case and description of the generic model MC-Model presented in Chapter 3, we have identified several challenging issues and specified security requirements. Accordingly, we identified the following design requirements:

R1: Authentication solution should provide entity authentication which,

- Can be mutual, or one-way from either side (client or cloud service),
- Should apply to all resource-access interactions,
- Should be provided for the duration of the user session (log-in to log-off).

R2: Authentication solution should support service access within different domains as well as service access within the same domain.

R3: Authentication solution should consider performance and reduce run-time delay and overhead introduced by END.

R4: Authentication solution should be elastic by adjusting the level of assurance.

R5: Authentication solution should be scalable. Scalability means the ability of END service to adapt to increased demands in order to accommodate more CSPs, services, and clients.

To satisfy the above requirements, we propose a novel an **E**lastic a**n**d **D**ynamic Authentication architecture, which will be referred to as END AuthN. END AuthN aims to provide authentication-as-a-service (AaaS) for all entities in M2C dealings. The design of END architecture has taken into account the following measures to satisfy requirements R1 to R5:

1- END architecture is designed as a stand-alone service that receives and handles authentication requests from any CSPs and services to authenticate clients and vice versa. This should allow for a one-way or mutual authentication if requested by interacting entities (R1). It will also make it possible for CSPs, services and clients from different domains to make use of the service (R2). In addition, this standalone cloud service will benefit from the scalability of the infrastructure, which is an underlying feature of CC (R4). It enables the service to expand or contract by adding or removing servers with minimal infrastructure- and software-level modifications [106].

2- END design applies the idea of elasticity by allowing service providers to decide the LoA requirements for the services they provide. END design can also adjust LoA through a variation of authentication Modes (R3).

5.8. Design Preliminaries

Following are the assumptions and definitions for END architecture:

A1: Each cloud resource (including services) has a trust requirement value that should be met in order to gain access to it. Meeting the trust value required is the basis for an authentication decision.

A2: CSPs decide trust requirements for accessing Cloud gateways and the individual services they offer.

A3: Mutual authentication is specified when interacting entities request authentication at both ends.

A4: When mutual authentication is specified, the trust requirement value at the resource side will be applied both ways, i.e., at the resource side and at the client side.

A5: Entity interactions can be captured from the control messages exchanged between entities.

A6: Trust values for all three categories (credential based, reputation based, and recommendation based) are all expressed on a scale from 1 to 4 in accordance with LoA scale suggested by NIST E-Authentication guidelines (See Appendix A).

5.9. Identification of Trust Factors

In order to quantify trust for an entity *e*, firstly, all trust factors that may contribute to setting the Level of confidence or Trust for ***e,*** need to be identified.

For the proposed END architecture, all factors associated with the entity whose trust requires evaluation must be considered. For MC-Model, we have two types of interactions requiring authentication decisions: user-to- system (or gateway), and service-to-service. For each of these types, we can identify different factors to guide the authentication decision. Based on the three modes of authentication identified in Section 5.6, trust factors are classified into three categories: credential-based referred to as **T.Cr** factors, reputation-based referred to as **T.Rp** factors, and recommendation-based factors referred to as **T.Rc** factors.

Definitions:

The following definitions provide the basic notation used in our work:

Def.1: Authentication Instance

An authentication instance is an authentication event that takes place in given computing environment for an entity in question e [102].

Def. 2: Trust Factors:

Trust factors refer to factors contributing to quantifying trust for a given entity in an M2C environment. Transaction history and other entities' recommendations are examples of these factors. Trust factors need to be identified and then quantified to determine their contribution to the overall trust value of a given entity **e**. In our proposed scheme, we identify three trust factors based on the three modes of authentication we identified:

A) Credential-based factors, which are quantified as **T.Cr.**

B) Reputation-based factors will be quantified as **T.Rp.**

C) Recommendation-based will be quantified as **T.Rc**.

Def. 3: Trust Component:

For each trust factor identified in Def.2, the trust component is the trust value computed for that factor. For example, the trust factor **T.Rp** has a component value *V(***T.Rp***) = a.*

Def. 4: Trust Weight Vector

The Trust weight vector **(w.cr, w.rp, w.rc)** represents the degree to which T.Cr, T.Rp, and T.Rc respectively, contribute to the overall value of trust.

Def. 5: Aggregate Trust (AggT)

The Aggregate Trust (denoted by **AggT**) is the overall value of trust determined by combining the contributions of all trust factors for a given entity **e**. So the overall trust for **e** is determined using the following equation:

$$AggT = (V(\mathbf{T.Cr}) \times w.cr) + (V(\mathbf{T.Rp}) \times w.rp) + (V(\mathbf{T.Rc}) \times w.rc) \qquad Equation\ (5-2)$$

The following sub-sections will define trust factors for the proposed architecture as they relate to the above-mentioned modes of authentication. Sub-Section 5.9.1 defines Level of Assurance (LoA) as a trust factor. Sub-sections 5.9.2 and 5.9.3 describe M2C entities' Reputation as a Trust Factor, and recommendation related factors for M2C entities as a trust factor, respectively.

5.9.1. Level of Assurance (LoA) as a Trust Factor

LoA and trust are related terms that play a major role in our proposed authentication scheme. The US National Institute of Standards and Technology (NIST) defines LoA as the degree of

confidence in the process of establishing the identity of an individual to whom an authentication credential was issued and the degree of confidence that the individual using the credential is the same as the individual it was issued to [16]. Hence all factors related to the process of authentication affect LoA including the method used for identity proofing, the types of credentials, and how credentials are managed [16]. In previous work, LoA-based authentication was investigated based on credentials for user-to-system authentication in a Grid environment [101], [102]. LoA was defined as the strength of authentication required to assure a service provider that access is granted to users whose identities have been verified [101]. For M2C, we expand this definition to be the strength of authentication required for a transaction to assure a service or a client or both that the identities of entities interacting with them are verified. This allows for mutual authentication based on the LoA required by CSPs, services or clients. It also entails examining system-to-system authentication, as in the case of a service requesting access to another service on behalf of a client.

One of the design goals of the current research is to use resources in M2C dynamically and effectively in a distributed environment. High trust values are only necessary for highly important tasks requiring access to more sensitive entity's data and processes. Less sensitive interactions may require less stringent trust values. This, in effect, relates trust to the LoA necessary for a given Authentication instance, where more or higher valued LoA factors lead to a higher trust evaluation of an entity. Hence, the aggregated value of credential based LoA factors is necessary for authenticating an entity and is one of the trust factors that will be considered in setting the overall trust for a given entity.

Chapter 6 is dedicated to the identification and analysis of credential based LoA related factors. It also proposes algorithms for combining these factors and quantifying a composite value for these factors, which will be denoted as ALoA.Cr = T.Cr.

5.9.2. Reputation as a Trust Factor

For our proposed architecture, we define reputation as the trustworthiness of an M2C entity (evaluated entity) based on its direct interaction history with evaluator entities. Hence, entities behaviour is logged in the database with every interaction. Behaviour is estimated through defined metrics. Examples of metrics may be used in M2C are freshness of last interaction previously recorded, or trust value required of services accessed for interaction previously

recorded. Freshness would be measured with a decay factor to limit the effect of old interactions. The trust value of previously accessed services would hint to the level of confidence in an entity previously authenticated for accessing the service; so if user Bob last interaction was with service S1, where T(S1) = 4, it would indicate high level of confidence in Bob to allow him access to a service with very restricted security level.

5.9.3. Recommendation as a Trust Factor

As opposed to reputation factor, recommendations are based on indirect relation with the entity in question. For END AuthN, we define recommendation as the trustworthiness of an evaluated entity e based on the history of relations between e and other entities in the M2C environment under consideration. Accordingly, entities behaviour is logged in the database with every interaction through defined metrics. Similar to metrics used for reputation, examples of metrics may be used for recommendation are freshness of previously recorded interactions, trust value required of services accessed for interaction(s) previously recorded. Another possible metric for estimating trust based on recommendations is the number of evaluators in previously recorded interactions since it means more entities attest to the behavior of e.

5.10. General Description for END AuthN Architecture

The goal of our novel M2C authentication architecture **END AuthN** is to decide whether an entity is what it claims. The system should output an authentication decision (i.e., Authentic or not). Since CC is a service-centered paradigm geared towards better resource utilization, the proposed solution provides authentication as a service. It is a standalone service that receives and handles authentication requests for CSPs, services, or clients. END service will have a centralized entry point for handling requests, which we will refer to as END-C. In order to carry out all operations necessary to output the authentication decision, potential users of END services (CSPs, services, or clients) will have an END peripheral component, which will be referred to as END-P.

Whenever an entity **e** requests access to service **S,** the required value of trust for S, **T(S)** as specified by the provider of service S. The mission of END is to identify and gather all trust-related factors and determine the three trust components *V(***T.Cr***), V(***T.Rp***), and V(***T.Rc***)*, which

in turn are used to compute AggT. If AggT matches T(S) for the transaction in question, then a positive authentication result is achieved. Figure 5-3. shows the general architecture of the END AuthN Service.

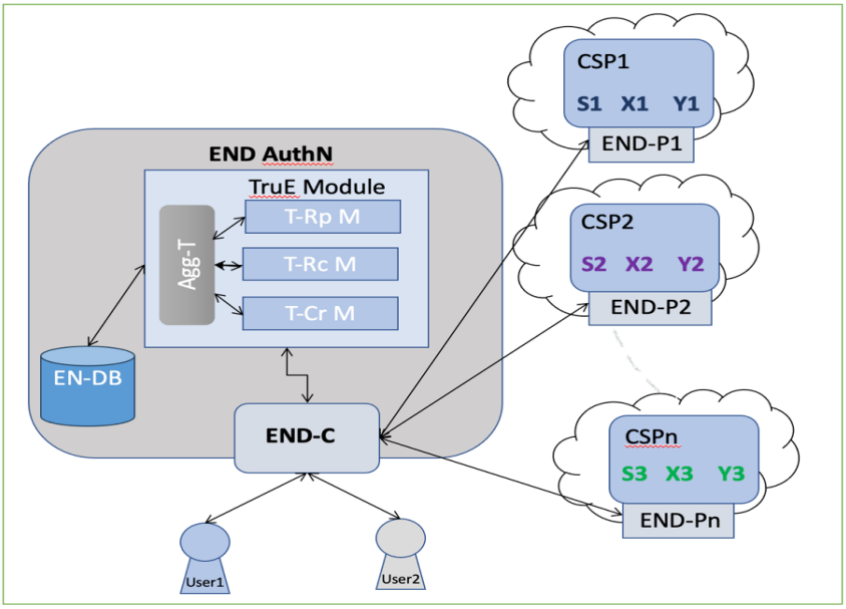

Figure 5-3. END Authentication Architecture

5.10.1. Architectural Components of END

In this section, we will describe in detail each component in the proposed END AuthN architecture (Table 5-2). The description will include the function of the component, the process utilized to perform the function, and the inputs and outputs of the process.

Table 5-2: END AuthN Architectural Components

END AuthN	**E**lastic a**n**d **D**istributed Authentication Architecture	Function
TruE	**Trust E**ngine	Identification and quantification of trust based on 3 modes of authentication
T-Rp M	Reputation Trust Module	Quantification of trust based on Reputation
T-Rc M	Recommendation Trust Module	Quantification of trust based on Recommendation
T-Cr M	Credential based LoA Trust Module	Quantification of trust based on LoA (Credential based factors)
AggT M	Aggregate Trust Module	Determination of aggregated trust based on the three trust factors: credentials, reputation and recommendation
EN-DB	END Database	Database for data gathered on interacting providers, services, and clients

5.10.2. Trust Engine (TruE)

The Trust Engine Component, or **TruE**, is a novel trust evaluation framework in charge of determining a composite numeric value for the trust of entities within a given authentication instance. Its function is divided among the following Modules described in subsections 5.10.2.1 through 5.10.2.4.

5.10.2.1. The Reputation-based Trust Module (T-Rp M)

The reputation-based trust module (T-Rp M) quantifies the trust value based on the value of reputation-based factors in M2C and stores data in EN-DB. The T.Rp is based on parameters derived from the history of relations between the two entities interacting, or direct history. The

choice of these parameters is another opportunity to add one more dimension to the elasticity of the architecture. Our proposed architecture assumes that these history-based reputation metrics can be chosen based on the nature of the interaction or upon the definition of SLA specification of CSPs and the services they provide.

This module initializes the parameters used to estimate *V(***T.Rp***)* and initialize the values of these parameters for newly introduced entities if necessary. It also uses recorded parameter values after the conclusion of each authentication instance to update the reputation value of interacting entities in EN-DB. *V(***T.Rp***)* is set on a four-value scale (1 through 4) so it can be entered as an input to the AggT derivation equation (section 5.8).

5.10.2.2. The Recommendation-based Trust Module (T-Rc M)

The recommendation-based trust module (T-Rc M) quantifies the trust value based on value for recommendations-based factors of entities in M2C environment which get recorded in EN-DB. The parameters used for T-Rc can be the same ones measured in direct history relations (T.Rp factors). However, these parameters will be measured and recorded previously for interactions between any two entities in the M2C environment. Furthermore, these metrics can be chosen based on the nature of the interactions or per the definition of SLA specification of CSPs and services they provide.

T-Rc module initializes the parameters used to estimate *V(***T.Rc***)* and initialize the values of these parameters for newly introduced entities if necessary. It also uses recorded parameter values after the conclusion of each authentication instance to update the recommendation value of interacting entities. *V(***T.Rc***)* is set on a four-value scale (1 through 4) so it can be entered as an input to the AggT derivation equation (section 5.9).

5.10.2.3. The Credential-based Trust Module (T-Cr M)

T-Cr module is dedicated to analysis and identification and quantification of credential based LoA related factors. The output of this module is the aggregate value of all credential-based LoA related factors denoted by ALoA.Cr, which is equal to T.Cr. Chapter 6 will focus on description of the function of T-Cr module.

5.10.2.4. The Aggregate Trust Module (AggT M)

The aggregate trust module (AggT M) aims to derive the aggregate trust value for an entity in a given authentication instance. As explained in Definition 5, aggregate trust is the overall value of trust for an entity **e** determined by combining the contributions of all trust-related factors. 2

This module determines the weight vector (*w*.cr, w.rp, w.rc) for the three trust-related factors (**T.Cr, T.Rp, and T.Rc**) in order to calculate Agg-T. We propose two simple methods for setting the weight vector values (*w*.cr, w.rp, w.rc):

- Static weights: The weights are set statically in the initialization phase. An example of static weights are the following values (*w*.cr, w.rp, w.rc) = (0.5, 0.3, 0.2). These values reflect that credentials have the most impact on trust evaluation, while recommendations of other entities are the least impactful because credentials represent empirical evidence when reputation and recommendations don't. Also it minimizes the effect of false evaluations given by entities. The static weight alternative is an intuitive simple approach that would have no bearing on the performance of the module.
- SLA specification of CSPs: Another alternative is to initialize the weight vector per CSPs' specification. This alternative provides another dimension of elasticity, where providers can choose the factors, they deem more trustworthy for the calculation of aggregate trust. A weight of zero would cancel the effect of any factors deemed untrusted by providers. This alternative is also simple and won't affect the performance of the process.

Then the aggregate trust value is calculated based on equation 5-1 in Def 5:

$$AggT = (V(\mathbf{T.Cr}) \times w.cr) + (V(\mathbf{T.Rp}) \times w.rp) + (V(\mathbf{T.Rc}) \times w.rc)$$

Equation (5-1)

5.10.3. The END Database (EN-DB)

The EN-DB database is used to store the credential tree structure, reputation, and recommendation factors for communicating entities.

EN-DB contains six tables. The first table, referred to as the **EntityType** table, stores coding for the three types of entities interacting in M2C (CSPs, Services, Users). Secondly, the **Entities** table contains attributes for all entities in the defined M2C environment including: an END-ID,

the type code for the entity as per EntityType table, the name of the entity and the trust value required for the entity, and latest trust values established for the three trust factors defined T.Cr, T.Rp, T.Rc. Then, the **CrFactors** table, stores LoA-related factor IDs and component values. The CrFactors table contains the pool of nodes (possible credentials) for the Cr-S-tree. The **Cr-tree** table is the fourth table, and it stores the credentials' structure. The fifth table, **ENDlog** stores the parameters used to estimate entities behaviour during interactions including: interaction ID, timestamp of interaction, and the trust requirement for the interaction. Finally, **Interactions** table relating entities to interactions with three attributes: interaction ID, entity1-ID, entity2-ID. Figure 5-4. depicts relations among these tables.

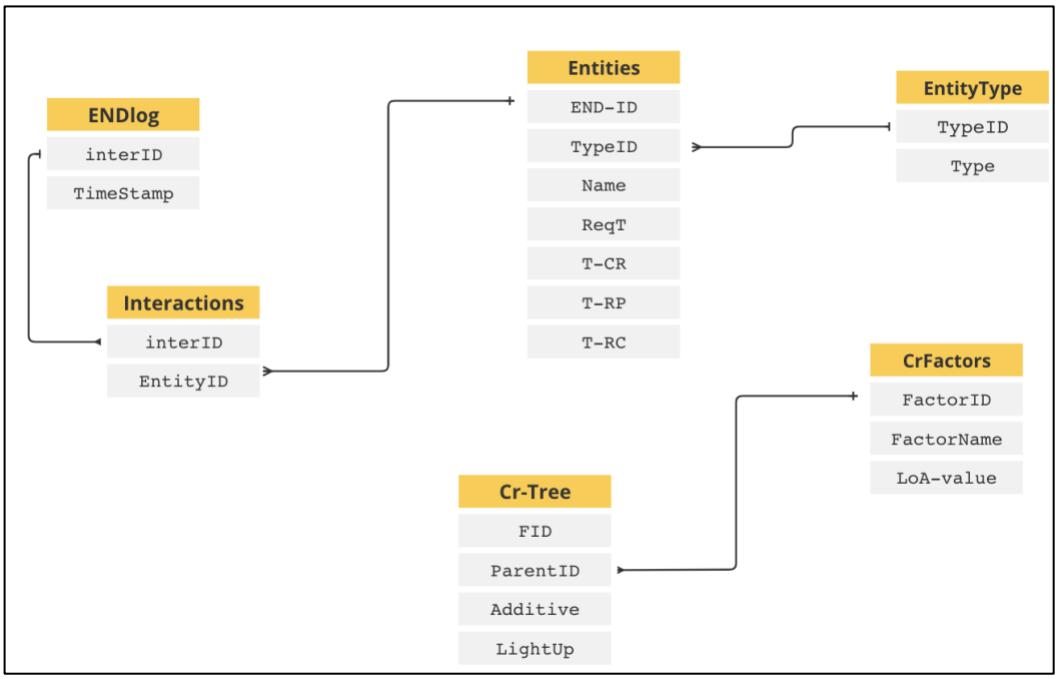

Figure 5-4. LoA-DB Tables and Relationship Diagram

5.11. Chapter Summary

This chapter reviews and analyses existing models for establishing trust in various solutions for computing environments comparable to M2C. Our analysis and observations of the various methods for establishing trust lay the groundwork for proposing a novel framework for establishing trust for interacting entities in M2C TruE. It also provides a general description of the novel proposed architecture (END AuthN) as well as a detailed description of its architectural

components, one of which is the trust framework. TruE framework depends on three trust factors, the first of which is LoA credential-based factors. The upcoming chapter defines and identifies LoA as one of the trust related factors and describes architectural components dedicated for this task.

Chapter 6: LoA as a Trust Factor: Design and Implementation

6.1. Chapter Introduction

In the previous chapter, we described the architecture of a novel elastic, dynamic, and distributed authentication for M2C environment END AuthN. It is based on three modes of authentication to establish trust in entities in the M2C environment: Reputation based, Recommendation based, and credential based LoA factors. The first two modes were detailed in chapter 5. Chapter 6 is dedicated to the Identification and quantification of credential based LoA-related factors. Section 6.2. presents definitions related to LoA-related factors, followed by Section 6.3 which identifies these LoA-related factors, while section 6.4 defines relationships between them and describes a novel framework for establishing trust for entities in M2C. Section 6.5 proposes a novel methodology for structuring these factors. Section 6.6 describes alternate algorithms for combining these factors to compute an aggregate value of trust based on these factors.

6.2. Definitions

More definitions related to our proposed architecture are presented in this Section. These definitions augment the list provided in Chapter (5) Section 5.8.

Def.6: Level of Assurance LoA for Service S

Level of Assurance LoA for service **S** is the degree of confidence in authentication necessary to access S based on credentials presented by a claimant in an authentication instance. It is denoted by **LoA (S)**, where

$$\mathbf{LoA}\ (\boldsymbol{S}) \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$$

Def. 7: LoA-related Factors

LoA-related factors refer to Authentication factors that contribute to setting the value of an authentication level of assurance associated with an authentication instance. Credential types or Cryptographic mechanisms used in the authentication process are examples of these factors.

The LoA-related factors, denoted as LoA.Cr factors, need to be identified, and then quantified to determine their contribution to the overall LoA needed for the Authentication instance.

Def.8: Level of Assurance Component

For **LoA.Cr** factor **a**, LoA component is the LoA value assigned for **a**. It is denoted by **LoA.Cr (a)**. For instance: LoA.Cr (PWD)= 1 means that the level of assurance for passwords is valued at 1.

Def.9: Aggregate LoA for Credential-based Authentication Factors

The Aggregate LoA for Credential based authentication factors (denoted by **ALoA.Cr**) is the overall value of LoA determined by combining the efforts of all LoA.Cr factors affecting the authentication of an entity **e**.

So for given *n* LoA-Cr factors, (where $n > 0$), the set of LoA components $\{LoA(c_1), LoA(c_2), \ldots, LoA(c_n)\}$ and the overall LoA associated with an authentication instance is determined through the following equation:

$$ALoA.Cr = f(LoA(c_1), LoA(c_2), \ldots, LoA(c_n)). \qquad \textit{Equation (6-1)}$$

where f is a mathematical function applied on the set of factors to produce the aggregate value of LoA.

6.3. Identification of Credential based LoA-related Factors

The NIST Electronic-Authentication (E-Authentication) guidelines provide the most comprehensive set of LoA-related factors categorized under the following classifications [16]: Registration and Identity proofing, Authentication Tokens, Token Mechanisms, and Credential Management, Authentication process protocols, and Assertions. LoA components are the values of LoA-related factors. NIST E-Authentication Guidelines describe four levels of assurance [16]: Level 1, Level 2, Level 3, and Level 4. These levels are defined based on the likely consequences of false authentication [16]. As these consequences get more serious, the required Assurance level gets higher. Since the requirements for each assurance level are incremental, assurance level 1 provides the least security, while assurance level 4 offers the greatest protection. Appendix A defines component LoA values for the above categories and describes requirements for each category at all LoA values as per NIST guidelines.

The authentication context (AC) specification defined in the SAML v2.0 standard specification set by OASIS has also identified a number of LoA-related factors (or Authentication Contexts, ACs). Authentication context is outlined as the additional information (other than the authentication assertion itself) that CSP may require to make a decision with respect to an authentication assertion [46]. ACs include five categories (with sub ACs under each) [62]: identification, technical protection, operational protection, authentication method, and governing agreement.

The work of Yao and others [101] built a comprehensive list of LoA-related factors combining features identified in NIST guidelines, OASIS specifications, and web security context working group W3C-SCWG web-related specifications. This work was proposed to provide effective authentication based on LoA for Grid environments. They also added the location of the user as a dynamic LoA related factor.

The current research will use the NIST four-level scheme. Appendix A lists component LoA values for the above categories and describes requirements for each category at all LoA values as per NIST guidelines [16].

For the M2C context represented by the M2C-Model described previously, three types of interactions that require authentication are identified (Figure 6-1): user-MC, and user-Service, and Service-Service, where Service is a Cloud service of any type and services may be within the same domain or cross-domain. The first two types signify user-to-system authentication, while the third one represents system-to-system authentication.

There are varying models depending on the entities involved in the authentication process: direct, simple assertion-based, and delegation (proxy-assertion) authentication. For user-to-system, all these models can be employed. However, system-to-system authentication may be limited to proxy (delegation) authentication. Yao and others have identified four scenarios for user-to-system in Grid environment [101]. However, we identify five possible scenarios for authentication for the MC-Model. The following scenarios consider user-to-system and system-to-system scenarios in M2C environment (illustrated in Figures 6-2 to 6-6):

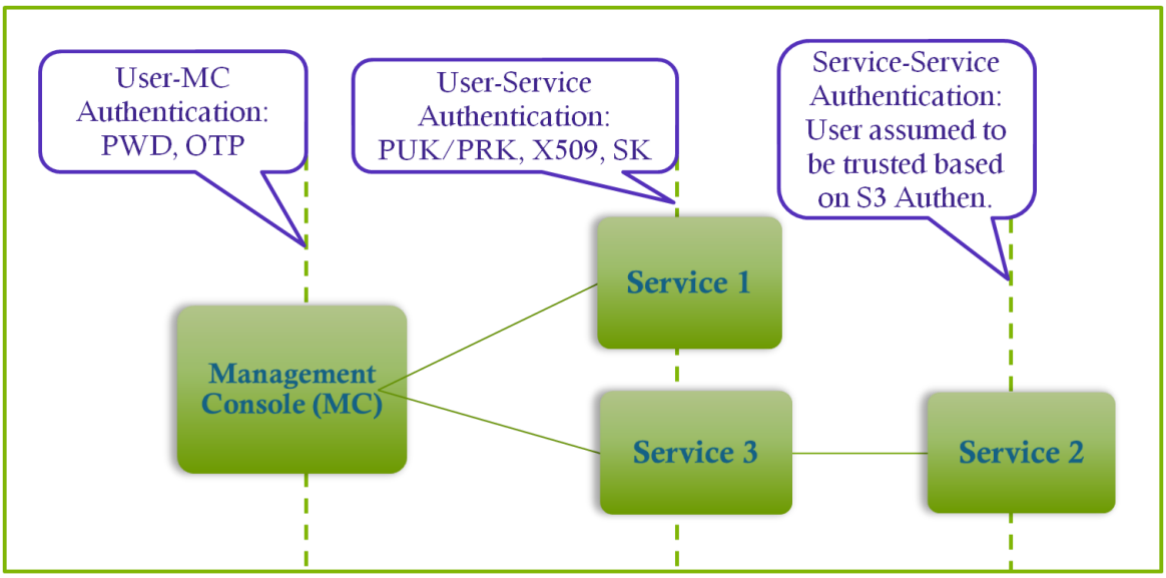

Figure 6-1. Types of Authentication in M2C Model

- **Scenario 1: User-to-System Direct Authentication scenario**

In this scenario, the user directly presents his token to authenticate with the Verifier, which in this case is a function of the CSP. As demonstrated in Figure 6.2, this scenario doesn't require Assertions. Credentials may be just a password or an OTP. This is the most generic and common scenario employed for user-to-system authentication in various environments. For the M2C Model, it is used for User-MC interactions.

Figure 6-2. User-to-System Direct Authentication Scenario

- **Scenario 2: User-to-System Direct Assertion-based Authentication**

  In this scenario, the user uses his token to authenticate with the Verifier. After successful authentication, the Verifier needs to create an assertion since the Verifier and the CSP are not collocated. The assertion is then sent to the CSP for authentication. This is usually handled automatically by the user's browser. This scenario may be used in the case of making payment

for services offered by the CSP where the CSP requires verification for Credit Card payment through another Verifier such as the “Verified by Visa” service.

The assertion may be a Holder-of-Key Assertion, which contains a reference to a symmetric key, or a public key possessed by the user. This, in turn, allows the CSP to require the user to prove possession of the referenced secret, which, to a degree, proves his rightful ownership of the assertion. If the assertion is a Bearer Assertion, it does not provide a way for the user to prove the assertion’s rightful ownership. Figure 6-3. exhibits this scenario.

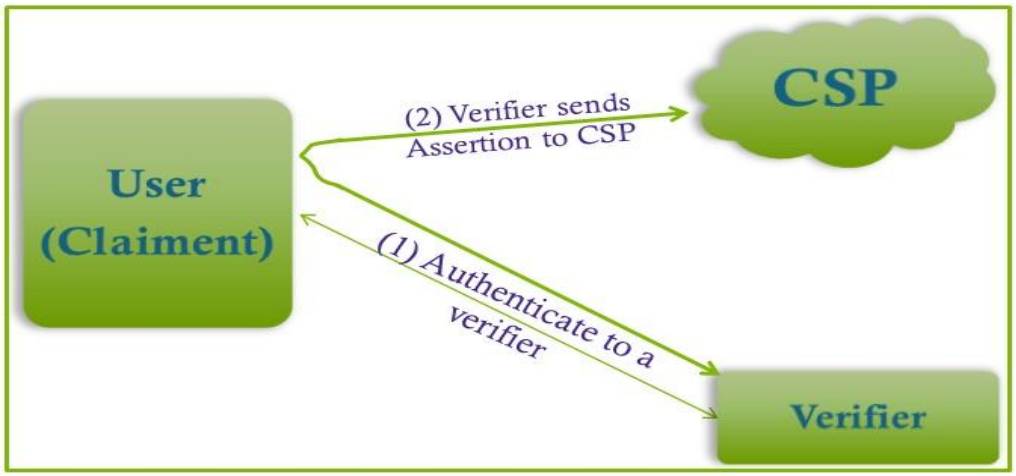

Figure 6-3. User-to-System Direct Assertion-based Authentication Scenario

- **Scenario 3: User-to-System Indirect Assertion-based Authentication**

Similar to the first two scenarios, the user uses his token to authenticate with the Verifier. After successful authentication, the Verifier creates an assertion and an assertion reference to identify himself and holds a pointer referring to the full assertion. The assertion reference is then sent to the user to be forwarded to the CSP for authentication. The CSP later uses the assertion reference to explicitly request the assertion from the Verifier. Figure 6-3. shows this sequence. The assertion may be a Holder-of-Key Assertion or a Bearer Assertion.

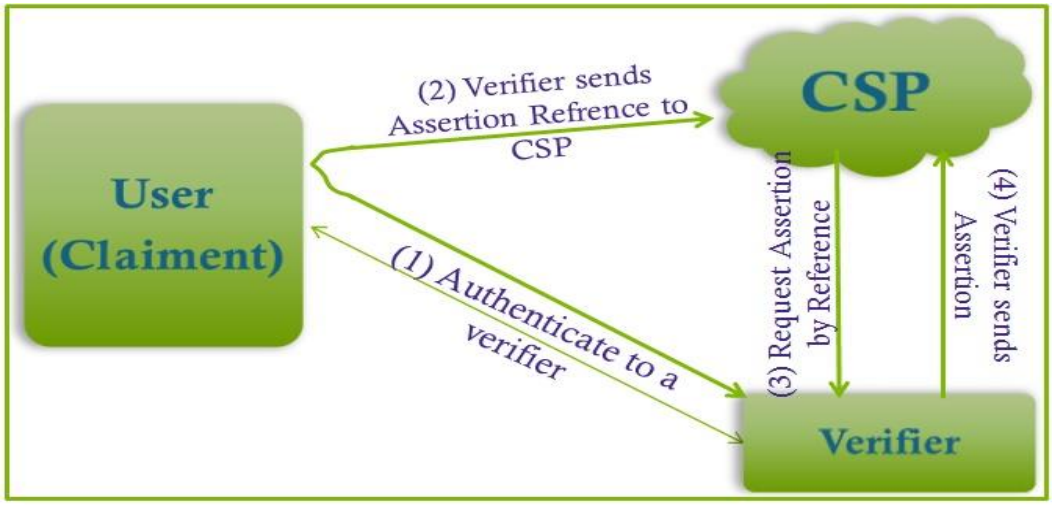

Figure 6-4. User-to-System Indirect Assertion-based Authentication

- **Scenario 4: User-to-System Proxy Assertion-based Authentication**

In this scenario, the user uses his token to authenticate with the Verifier. Following successful authentication, the Verifier acts as an intermediary and interacts directly with the CSP. Hence, the assertion is directly sent from the Verifier to the CSP. The CSP grants or denies the authentication request based on the assertion made by the Verifier. Proxy assertions are useful in cases where users request access to multiple service providers or for network monitoring and filtering mechanisms. Typically, communications between the Verifier and the CSP are protected by the proxy employing client-authenticated TLS with the Verifier and passing the authentication assertion in the HTTP header [16]. This is shown in following Figure 6-4.

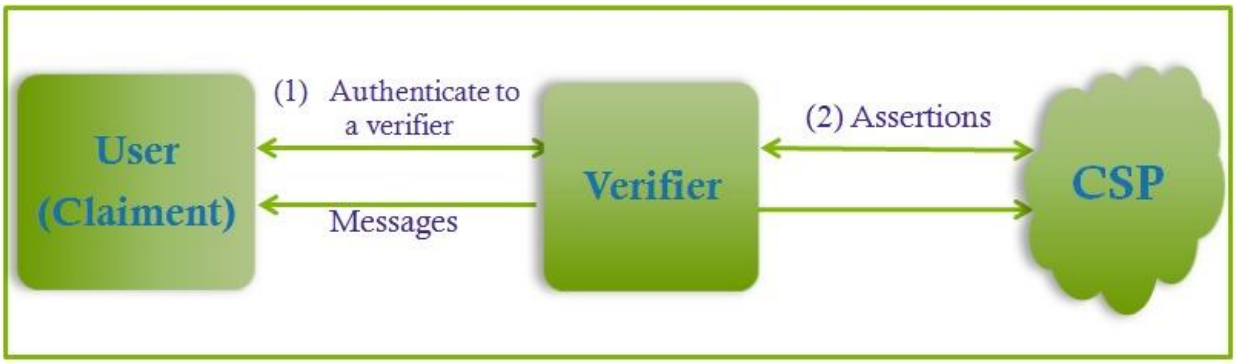

Figure 6-5. User-to-System Proxy Assertion-based Authentication scenario

Cookies, Security Assertion Markup Language (SAML) Assertions and Kerberos Tickets are examples of proxy Assertions.

- **Scenario 5: System-to-System Assertion-based Authentication**

In this scenario, a system component such as a cloud service needs to authenticate with another system component, which may be another cloud service. Both cloud services may be in the same domain or have different domain as in the case of services offered by different CSPs. Assertion-based authentication enables Single-Sign-On for entities, allowing them to authenticate once to a verifier and then obtain services from multiple service providers without requiring additional authentication. As explained before, a user uses a token to authenticate to the verifier. Following successful authentication, the verifier acts as an intermediary and interacts directly with the CSP. Hence, the assertion is directly sent from the verifier to one or more CSPs on behalf of the authenticated user. To protect communications between the Verifier and CSPs, the proxy uses client-authenticated TLS with the Verifier and passes the authentication assertion in the HTTP header [16]. Figure 6-6. expresses this scenario.

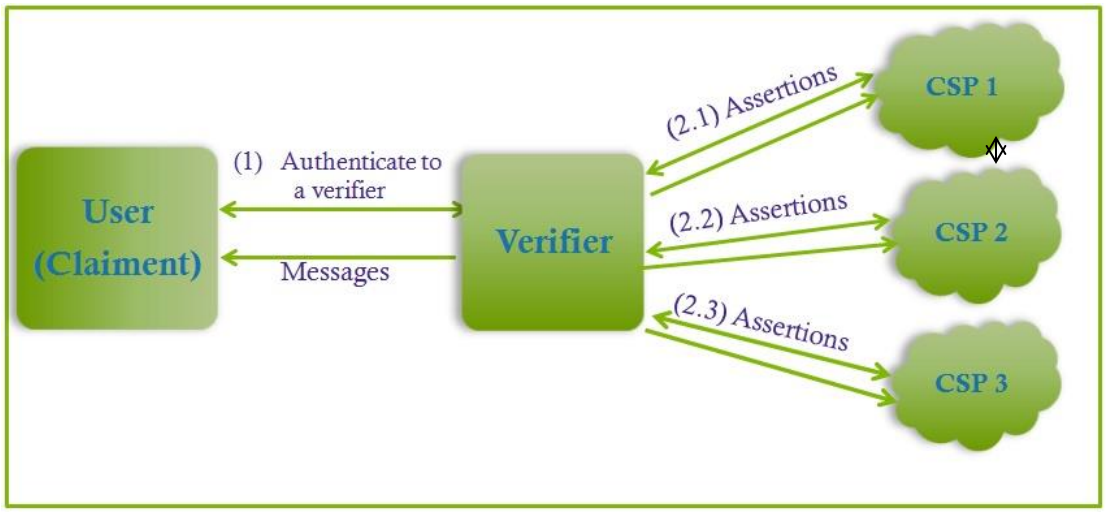

Figure 6-6. System-to-System Assertion-based Authentication Scenario

6.4. Defining Relations Between Credential based LoA-related Factors

The overall LoA is affected by the relations among the entities, as described in the scenarios presented in Section 6.3, and how authentication is performed. So, the following rules are applicable to determine the overall A*LoA.Cr* :

1. The first rule is the low watermark rule, where the lowest LoA is chosen as the overall LoA [16], [101], [103]. The low watermark rule applies when factors are co-dependent or mutually dependent on the same set of circumstances. This can be seen in the following situations:

- From the presented scenarios, we can see that an authentication instance between two entities is comprised of two or more authentication events. For these scenarios, A*LoA.Cr* is based on the low watermark of the LoA values for each of the component authentication events of the instance [102], [101]. To understand the low watermark rule, we present the following example: If an authentication instance is comprised of two steps (Step1 and Step2) with LoA values *LoA(Step1)* = 2 and *LoA(Step2)* = 3, then

  $ALoA.Cr = min\ (LoA(Step1)\ ,\ LoA(Step2)\ ) = 2$.

- Each authentication instance has several components from the various categories defined by NIST [16], which are R and IdP tokens, token and credential management, authentication protocols, and assertions. So, for any authentication instance, A*LoA.Cr* depends on factors from all these categories. The low watermark of the LoA values for each of these components also applies. For instance, if we have an authentication instance with component values as follows:

-The token employed is MF Software Cryptographic Token, resulting in *LoA(token)* = 3.
-The authentication protocols used have *LoA(protocol)* = 3.
-The token and credential management processes used have *LoA(T and CM)* = 2.
-Authentication assertions were not used.
-Then, the low watermark rule is employed, resulting in:

A*LoA.Cr = min (LoA(token), LoA(protocol, LoA(T and CM)) = 2*.

- Another situation in which this rule is applied is in the multi-stage authentication processes [16], where a single-factor token is used to obtain a second token. The A*LoA.Cr* associated with

the compound solution has a LoA value equal to the weakest component value of both tokens. When a cryptographic solution permits full or partial cryptographic keys to be stored on an online server and downloaded to the user's local system following successful authentication with a passphrase is one example. Then, the user can authenticate to a remote Verifier using the downloaded software cryptographic token. The overall authentication process is considered only as strong as the passphrase used to obtain the cryptographic token.

The rationale behind the low watermark rule is that the point with the lowest level of assurance will likely be the target of an Attacker. So, the factor with the lowest LoA component value will bring the composite value down to its level. For example, if a system uses an authentication token with a LoA value of 2 but employs other mechanisms with a LoA value of 3, an Attacker will likely focus on exploiting the token since it is easier to attack a component with a lower LoA.

The additive rule is applied when several independent factors contribute to the aggregate LoA [103]. It also applies in cases where factors are not completely independent, but each factor may depend on different circumstances. So, the overall A*LoA.Cr* should be higher, and the security is stronger when two types of tokens, such as username/password and OTP, are employed rather than just passwords.

Understanding these relations is the first step which enable us to structure all LoA-related factors and arrive at an overall A*LoA.Cr*.

6.5. Structuring Credential based LoA-related Factors

After defining the rules that govern possible relations between LoA-related factors, we can organize these factors in a manner that allows us to systemically apply these rules and deduce a composite value for A*LoA.Cr*.

For a given authentication instance, LoA-related factors are structured in groups according to the governing rules of their relations (low water mark or additive) at various levels. To arrive at this structure, various decisions need to be made regarding the factors examined. Figure 6-7. shows the flow chart for this decision process.

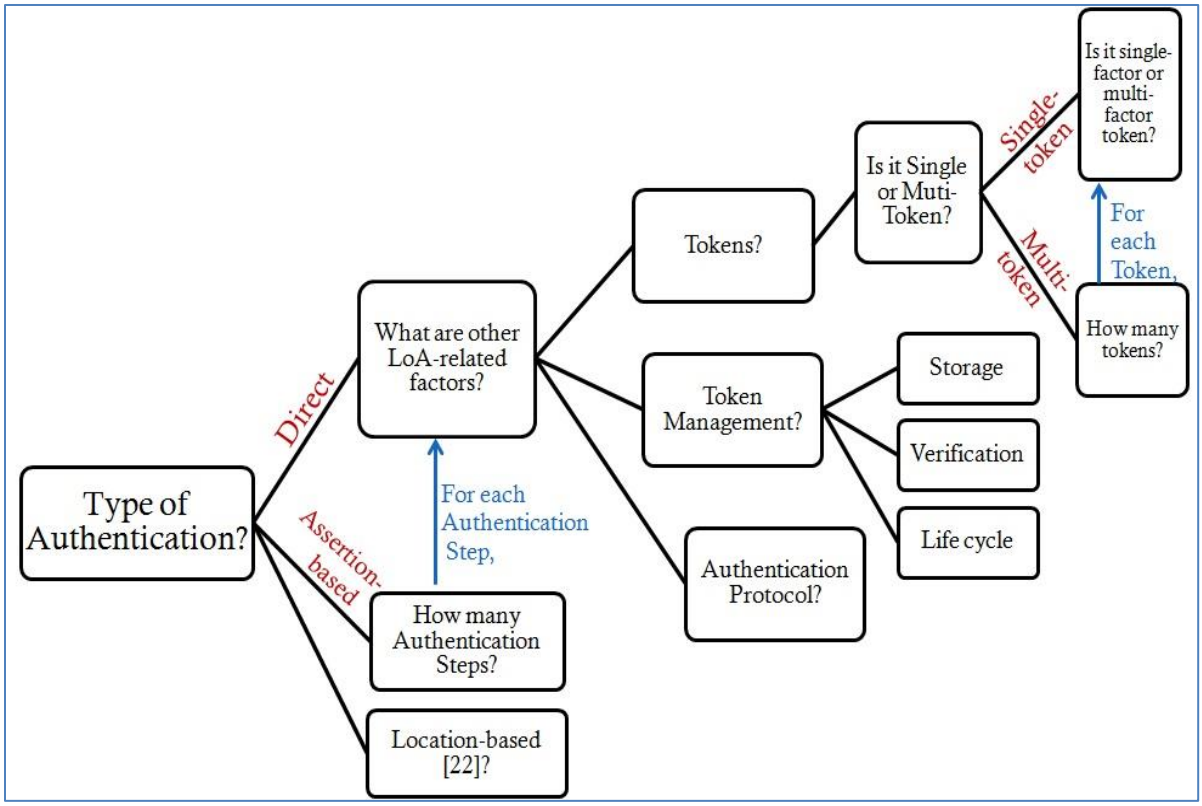

Figure 6-7. Decision Tree to Build a Structure for LoA-related Factors

Following the above Decision tree, we can build a tree structure of factors for a given authentication instance. We can refer to the resulting structure as the "LoA-factor Tree". Figure 6-8. offers an exemplar LoA-factor Tree, which shows all LoA-related factors structured into groups arranged at varying levels; each level conjoins related factors into groups, allowing us to identify the appropriate rule (low water mark or additive) to apply. To deduce an overall LoA, we need to traverse the LoA-factor Tree. Starting at the leaves and going up the Tree, for all nodes with the same parent, we apply the suitable rule, arriving at a composite LoA value to be passed to the parent.

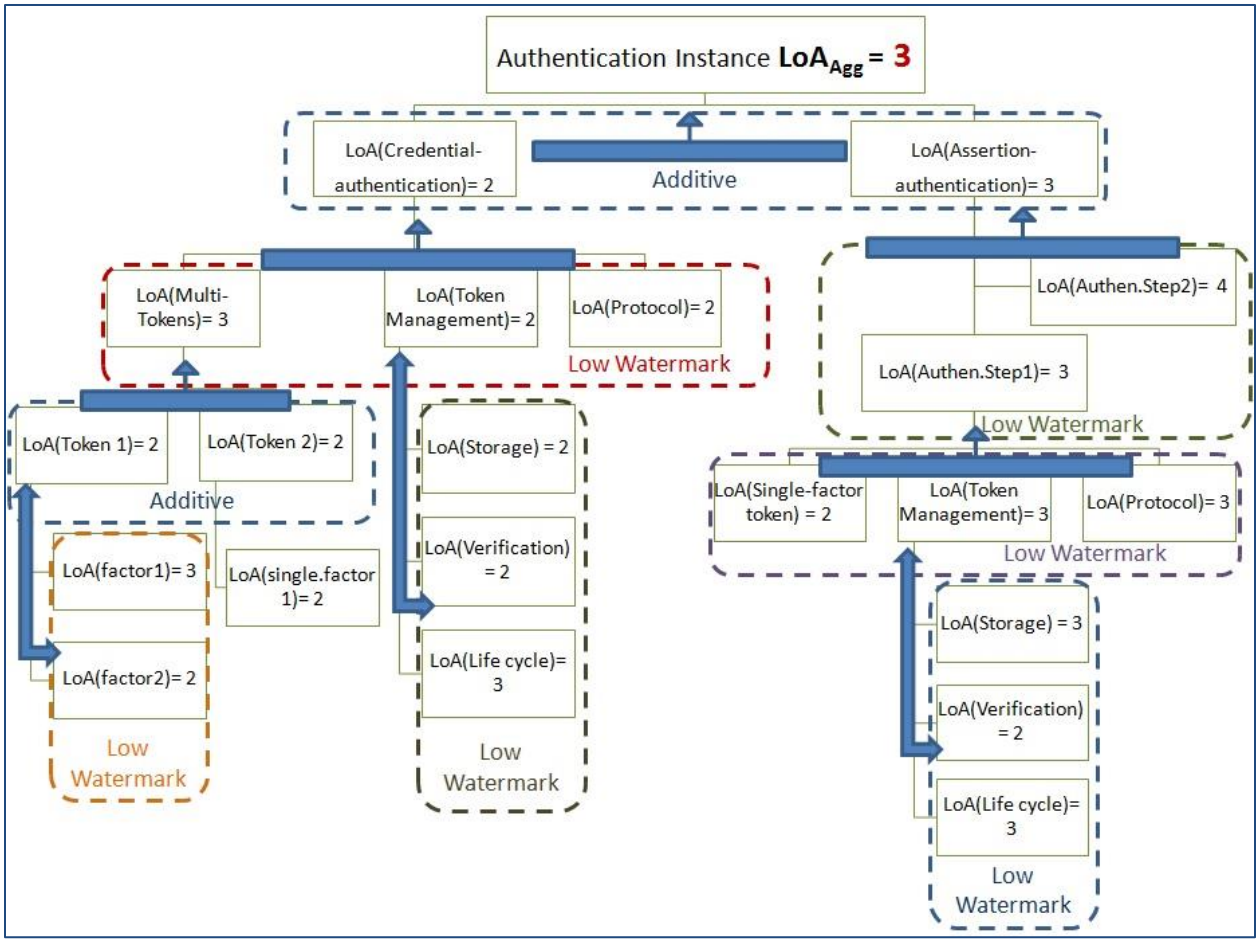

Figure 6-8. An Exemplar LoA-factor Tree for a Given Authentication Instance

6.6. Determining Aggregate value for credential based LoA-related factors

Given the following definitions:

- $ALoA.Cr$ = The aggregate value for LoA considering a group of LoA-related factors relating to a given authentication instance,
- LoA_{LW} = The composite value of LoA considering a group of LoA-related factors when applying the Low Watermark rule,
- And LoA_{Add} = The composite value of LoA considering a group of independent LoA-related factors when applying the Additive rule,

We define Equation 6-2 as:

$$ALoA.Cr = \begin{cases} LoA_{LW} = \min(R_L) \; where \; R_L \; = a \; set \; of \; LoA \; component \; values \; for \; LoA \; related \; dependent \; factors \\ LoA_{Add} = f(R_A) \; where \; R_A = a \; set \; of \; LoA \; component \; values \; for \; LoA \; related \; independent \; factors \end{cases}$$

Equation (6-2)

$ALoA.Cr$ is a recursive function, where the sets R_L, R_A may require applying the same equation at a different level of the LoA-factor Tree. The algorithm for calculating $ALoA.Cr$ is shown in Table 6-1. This algorithm assumes that the Tree structure has been defined and created.

Table 6-1: Algorithm for Calculating ALoA.Cr for a given Authentication Instance

Input: LoA-related factors Set $\boldsymbol{R}$ $|R|$ >1

Output: Aggregate value of LoA $\boldsymbol{ALoA.Cr}$

Integer Method $\boldsymbol{ALoA.Cr}$ (Object Set R)

```
    while Tree not empty
        If node is a leaf and has no siblings, then  // one factor in the group
            return LoA(node);
        else node is a leaf and has siblings, then   // apply proper rule for group of factors
            rule ← node.rule;
            save leaf and siblings into a new object Collection siblings[];
            If rule = Add, then
                ALoA.Cr ← LoA_Additive(siblings);
            Else
                ALoA.Cr ← LoA_LowWatermark(siblings);
        else                  // determine ALoA.Cr for node based on children factors
        save children of node into a new object Collection children[];
        ALoA.Cr (children);
    End while;
return ALoA.Cr ;
```

6.6.1. Algorithms for the Low Watermark Rule

In the case of dependency between LoA-related credential-based factors, the governing rule is the low watermark rule. The result, as shown in the ALoA.Cr general equation, is computed by applying the minimum function on the set of LoA component values.

$LoA_{LW} =$

$\min(R_L)\, where\ R_L\ a\ set\ of\ LoA\ component\ values\ for\ LoA\ related\ dependent\ factors$

We examine two methods for determining the Minimum value LoA. First approach for determining the minimum value is through an ascending sort of the array of LoA component values stored as presented in Table 6-2. The minimum value is then the first element of the array. Another method is through pairwise comparisons between elements of the array as shown in Table 6-2.

Table 6-2 Algorithm for Sort to Determine Minimum LoA Value

Input: LoA-related factors Set $\boldsymbol{R_L}$, $|R_L|$ >1

Output: Aggregate value of LoA $\boldsymbol{LoA_{Agg}}$

Integer Method LoA.LowWatermark(Object Set R_L)

while Set $R_L$not empty

read one LoA component value in R_L;

save into Integer ARRAY LoA[];

end while

SORT ARRAY LoA[];

$\boldsymbol{LoA_{Agg}} \leftarrow$ min(LoA[x]);

return $\boldsymbol{LoA_{Agg}}$;

Table 6-3: Algorithm for Pairwise Comparisons to Determine Minimum LoA Value

..

min ← LoA[0];

for i = 1 to |LoA|

if LoA[i] < min then

min ← LoA[i];

..

Comparative evaluation of these two alternatives is provided in Chapter 7.

6.6.2. Algorithms for the Additive Rule

The additive rule governs the independent LoA-related credential-based factors. The general equation for determining the composite value for $\mathrm{LoA_{Add}}$ for a set of LoA-related factors is:

$$LoA_{Add} = f(R_A)\ where\ R_A$$
$$= a\ set\ of\ LoA\ component\ values\ for\ LoA\ related\ independent\ factors$$

Yao and Zhang suggested two algorithms for computing LoA_{Add} for a given authentication instance: an algorithm based on subjective logic [102], and an algorithm based on the inclusion-exclusion principle of Probability Theory [103]. Evaluation of the performance of the algorithms shows that the algorithm rooted in Probability Theory offers better performance [103], [101]. In our work, we propose two more algorithms: weighted average Algorithm (which we refer to as $\mathrm{W.LoA_{Add}}$), and mapped weight algorithm (which we refer to as $\mathrm{M.LoA_{Add}}$). The two latter algorithms suggested in this work use LoA component values as the basis for weighting contributions (w_i) of each LoA-related factor in the formula:

$$\mathrm{LoA_{Add}} = \sum_{i=1}^{m} LoA_i \times w_i$$

Equation (6-3)

The logic behind our two algorithms is that factors with higher LoA provide higher level of confidence as contributors to an authentication decision and thus should have greater effect on the aggregate LoA value for a set of factors. We also proposed a third algorithm: The Combined Factors Algorithm (which we refer to as $\mathrm{C.LoA_{Add}}$). The following subsections will explore in detail these algorithms. Evaluation of these algorithms is provided in Chapter 7.

Probability Theory Algorithm

The performance evaluation of the two algorithms for calculating $\mathrm{LoA_{Add}}$ is suggested in previous work: $\mathrm{S.LoA_{Add}}$, and $\mathrm{P.LoA_{Add}}$ [103] showed that the algorithm rooted in Probability Theory offers better performance [103], [101]. Thus, we will describe the $\mathrm{P.LoA_{Add}}$ algorithm in order to evaluate its performance against our suggested algorithms. P (r_i) is defined as the probability of the trustworthiness of an LoA-related factor r_i, which signifies the component value of $\mathrm{LoA_i}$

[101]. P (r_i) is a value in the range [0, 1]. Table 6-4 provides a mapping from LoA four level schemes to probability values within the range [0, 1].

Table 6-4: Mapping from LoA component values to Probability values

LoA Component Values for (r_i)	**Corresponding P (r_i)**
1	0.25
2	0.5
3	0.75
4	1

Based on the inclusion-exclusion principle for **n** independent events (factors), we can determine the probability of the additive LoA using the following [101]:

$$P.\,LoA_{Add} = 1 - \prod_{i=1}^{n}(1 - (P(r_i)) \qquad \textit{Equation (6-4)}$$

The algorithm is presented in Table 6-5.

Table 6-5: Algorithm for Additive Rule based on Probability theory ($P.\,LoA_{Add}$) [101]

Input: LoA-related factors Set R_A, $|R_A|$ >= 1

Output: Aggregate value of LoA **ALoA.Cr**

Integer Method LoA.PAdditive(Object Set R_A)

read LoA component values in R_A;

save into Integer ARRAY LoA[];

if $|R_A| = 1$, then

return LoA[0]; // In case of one factor

else if $|R_A| > 1$ then // Otherwise, apply inclusion-exclusion principle

temp = 1;

for i = 0 to n-1

```
                    temp = temp * (1- w[i]); //w =P[r] /4
                temp = (1 –  temp) ;
        ALoA.Cr ← temp;
    return ALoA.Cr;
```

Weighted Average Algorithm

The first algorithm we propose is a weighted average of LoA component values, or $\mathrm{W.LoA_{Add}}$. In this case, the value for the weight of each factor in equation (6-5) is represented by:

$$w_i = {}^{LoA_i}\Big/_{\sum_{i=1}^{m} LoA_i} \quad \textit{Equation (6-5)}$$

Applying this formula for w_i assures that highly trusted LoA-related factors pledge greater contribution to the aggregate *ALoA.Cr*. However, this formula can't be used for our purpose as it implies a dependency between the different factors since the weight for each factor w_i depends on the values of all other factors $\sum_{i=1}^{m} LoA_i$. This dependency will possibly lead to *ALoA* value that is less than some of the component values for the higher contribution factors. In other words, adding more LoA-related factors when making an authentication decision would not improve the trust level of a given entity in our scheme. The following numerical example will illustrate this effect. For n factors, we have R = {r_1, r_2, …., r_n}, and $\mathrm{R}_A = \{\mathrm{LoA}_1, \mathrm{LoA}_2, \ldots., \mathrm{LoA}_n\}$. Suppose n =3, with R = {f_1, f_2, f_3} and $\mathrm{R}_A = \{1, 2, 3\}$. Then application of the weighted average algorithm will yield the following results:

$$\mathrm{W.LoA_{Add}} = \sum_{i=1}^{3} LoA_i \times w_i = \left(1 \times \frac{1}{6}\right) + \left(2 \times \frac{2}{6}\right) + \left(3 \times \frac{3}{6}\right) \approx 2.33$$

In this example, the highest component value for a factor is $\mathrm{LoA}_3 = 3$. Application of the weighted average algorithm yields a lower LoA value. Hence, utilizing the third factor f_3 without the added contribution of factors f_1 and f_2 would be sufficient for making an authentication decision. Thus, this algorithm doesn't serve the goal of the additive rule and will not be evaluated.

Mapped Weight Algorithm

As explained before, using $\mathrm{W.LoA_{Add}}$ to calculate $\mathrm{LoA_{Add}}$ creates dependency between the factors, which are presumably independent. So, we propose another algorithm also rooted in the idea that factors with higher LoA component values should have higher weights in equation (6-6).

We refer to this algorithm as the mapped weight algorithm ($\mathrm{M.LoA_{Add}}$). Hence, we propose to use a similar mapping to that presented in Table 6-4.

- If $\mathrm{LoA_i}$ =1, then mapped w_i = 0.25,
- If $\mathrm{LoA_i}$ =2, then mapped w_i = 0.5,
- If $\mathrm{LoA_i}$ =3, then mapped w_i = 0.75,
- And If $\mathrm{LoA_i}$ =4, then mapped w_i = 1.

Hence the aggregate LoA is calculated by the following:

$$\mathrm{M.LoA_{Add}} = \sum_{i=1}^{n} LoA_i \times \ mapped\ w_i$$

Equation (6-6)

So, for the same numerical example presented in Section 6.6.2, the aggregate LoA for the three independent factors is calculated as:

$$\mathrm{M.LoA_{Add}} = \sum_{i=1}^{3} LoA_i \times \ mapped\ w_i \ = (1 \times 0.25) + \ (2 \times 0.5) \ + \ (3 \times 0.75)$$

$$= 0.25 \ + \ 1 + \ 2.25 = 3.5$$

This equation is simplistic and requires few calculations. It also sustains the independence of the factors and applies higher weights to more trusted factors. The algorithm is presented in Table 6-6.

Table 6-6: Algorithm for Additive Rule based on Mapped Weights

Input: LoA-related factors Set $\mathrm{R_A}$, $|\mathrm{R_A}|$ >= 1

Output: Aggregate value of LoA ALoA.Cr

Integer Method LoA.MAdditive(Object Set $\mathrm{R_A}$)

read one LoA component value in $\mathrm{R_A}$;

save into Integer ARRAY LoA[];

read attribute weights in $\mathrm{R_A}$;

save into Integer ARRAY weight[]; // weight = LoA/4

if $|\mathrm{R_A}| = 1$, then

```
                return LoA[0];      // In case of one factor
        else if |R_A| > 1 then
                  variable index ← 0;
                  variable temp ← 0;
                repeat
                            // Adjust b values with weights
                        LoA[index] = LoA[index]* weight[index];
                    temp ← temp + LoA[index];                  // calculate the ALoA.Cr
                      index ← index+1;
                until index = |R_A|;
        ALoA.Cr ← temp;
return ALoA.Cr;
```

Combined LoA Algorithm

The combined LoA algorithm is based on a scheme similar to that described for combining authentication tokens to achieve higher LoA. We refer to this algorithm as $C.\,LoA_{Add}$. According to NIST electronic authentication guidelines, each of the token types may be used to achieve a certain LoA when used in a single-token authentication scheme [16]. When combining two token types for a multi-token authentication scheme, a higher LoA can be achieved by the combination. Thus, combining two tokens with a LoA of *x* would achieve a LoA of *x+1*, where $x \in \{1, 2, 3, 4\}$. In a similar manner, combining two factors with LoA of *x* would achieve LoA of *x+1*, where $x \in \{1, 2, 3, 4\}$ in our scheme. This can be applied in a recursive manner to achieve a higher LoA. To illustrate the scheme, we present the following numerical example. For n factors, we have $R = \{r_1, r_2, \ldots, r_n\}$, and $R_A = \{LoA_1, LoA_2, \ldots, LoA_n\}$. Suppose n =4, with $R = \{f_1, f_2, f_3, f_4\}$ and $R_A = \{1, 1, 2, 2\}$. So combining f_1 and f_2 would allow us to achieve LoA of 2. And combining the three factors f_1, f_2 and f_3 would allow us to achieve LoA of 3.

This method sustains the independence of the factors and is also intuitive. The algorithm is presented in Table 6-7.

Table 6-7: Algorithm for Additive Rule based on Combined LoA

```
Input: LoA-related factors Set R_A, |R_A| >= 1
Output: Aggregate value of LoA ALoA.Cr
Integer Method LoA.CAdditive(Object Set R_A)
        read one LoA component value in R_A;
                save into Integer ARRAY LoA[];
         Integer ARRAY Count[];      // count of LoA equivalent LoA component values
        if |R_A| = 1 , then
                return LoA[0];      // In case of one factor
        else if |R_A| > 1 then
                  variable CLoA ← 1;
                  variable count array ← 0;
                for index = 1 to 3     // for LoA component values to be possibly combined
                (1 to 3)
                     for x = 0 to |R_A|
                          if LoA[x] = index then
                                count[index] = count[index] +1;
                      if count[index] >= 2 then     // when 2 or more factors have LoA = index
                           Count[index+1] = 1;
                           CLoA = index+1;
        ALoA.Cr ← CLoA;
return ALoA.Cr;
```

6.7. Description of architectural components related to LoA credential-based factors (T-Cr M)

In section 5.9, we describe components in the proposed END AuthN Architecture. TruE framework is the main component in charge of determining the AggT based on three modes of authentication. In this section, we will describe the structure and functionality of the T-Cr Module, which is part of the TruE framework. The T-Cr Module is dedicated to the analysis, identification and quantification of credential based LoA related factors. The output of this module is the aggregate value of credential based LoA related factors denoted by ALoA.Cr, which is T.Cr. The sub-components of T-Cr module are listed in Table 6-8.

Table 6-8: Sub-Components of T-Cr Module

Component Abbreviated Name	Component Functionality
CRSTM	Construct LoA.Cr static (base) tree and store into EN-DB
CRLTM	determine credentials light-up tree based on credentials identified for current authentication request
ALoAM	Compute ALoA.Cr based on derivation rules

The next sections are organized as follows: Section 6.7.1. describes a working example for END AuthN to show its Execution flow. Sections 6.7.2 through 6.7.4 will provide detailed description of the sub-components of T-Cr module listed in Table 6-8 above.

6.7.1. END AuthN Execution Flow

In order to understand how the END authentication service works, let's consider a client Bob, who is a registered user with CSP1, and CSP2. Bob needs a compute service from CSP1 to execute his code. All his data is stored on a storage service provided by CSP2. CSP1 has deployed an instance of the END service to handle authentication requests for CSP2 services and clients. A set of offline steps takes place first. These off-line steps are executed upon deployment of an instance of END service and are:

1) The base static credentials tree is constructed, which is the function of the LoA credentials static tree Module (CRSTM).
2) All of the information of the static tree is stored in EN-DB
3) The corresponding component LoA values are stored as well.

The execution flow of the END AuthN service to fulfill the authentication needs for these interactions are as follows. (Refer Figure 6-9).

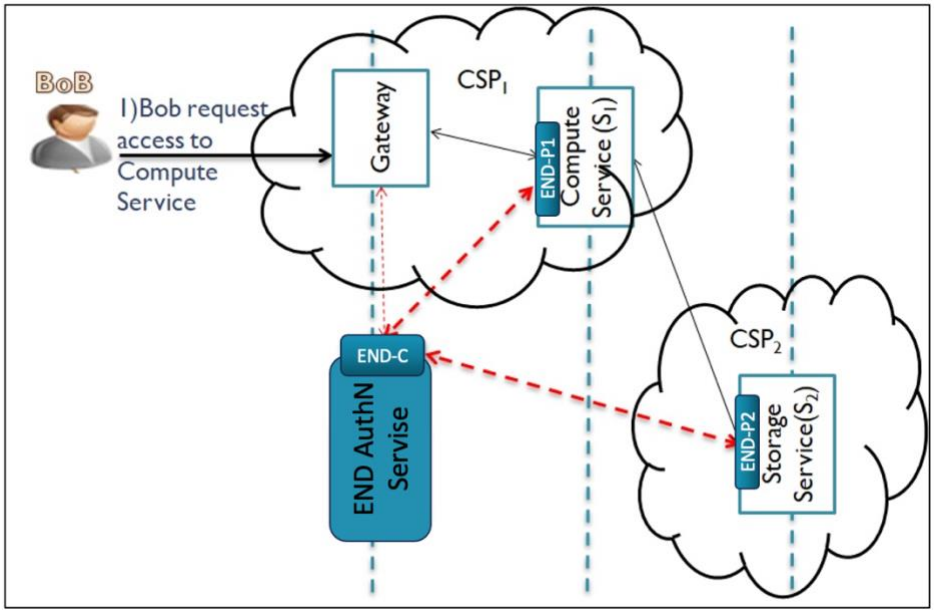

Figure 6-9. END AuthN Working Example

4) Bob requests access to Compute Service S1 from the management console webpage of CSP1. Mutual authentication isn't specified. END-P1 invokes the END AuthN service. The TruE Module determines AggT for this authentication instance based on the information gathered. If AggT value meets the trust requirement for S1 or *T(S1)*, Bob is authenticated and granted access to S1. Information gathered includes credentials Bob offered, reputation (based on stored direct-history between S1 and client Bob), and recommendations from other users or services available to END.

5) S1 requests access to data stored on S2 storage service. END-P2 invokes the END AuthN service. TruE Module determines AggT for this authentication instance based on information

stored in EN-DB. If the AggT value meets the trust requirement for S2 or *T(S2)*, S1 is authenticated and granted access to S2 on behalf of Bob.

6) At the end of the session, information collected about the session is used to update END databases to be used for future authentication requests.

6.7.2. The Credentials Static Tree Module (CRSTM)

The credentials static tree module (CRSTM) aims to construct credentials static (base) tree off-line and store it in EN-DB. This module is responsible for identifying a set of LoA-Cr factors in a given authentication environment, and then constructing a tree of all credentials collected, which will be referred to as the LoA credentials static tree CR-S-tree. The construction of the Cr-S-tree is based on grouping these factors based on their mutual relationships. We assume that the task of identifying these factors and grouping them is completed manually by an administrator or a security policy manager based on security policies and system requirements. This manual task is guided by the decision tree presented in Section 6.5. The tasks to be undertaken by CRSTM module are:

- The first task depends on the authentication environment and security policies. Built on investigating, credential-based authentication-related factors and the examination of the mutual relationships among these factors, a Cr-S-tree such as the one presented in Figure 6-7. is constructed. The figure shows that factors assembled at the same horizontal level have one of two types of mutual relationship, additive relationship, or lowest watermark relationship. The component value of a group of factors can later be determined based on applying the additive rule or low-watermark rule detailed in Section 6.6.
- Another task is the maintenance of the Cr-S-tree, which is accomplished by doing all necessary modifications, including adding, deleting, or classifying LoA.Cr factors, based on input from an administrator.

The algorithm for constructing the static tree is as follows:

Input: LoA-related decisions Set ***D*** with related options and LoAs (Figure 6-7.)

Output: Credential base tree

```
/* List of decision objects where each object contains a decision Q, No of possible
   Decisions, and pointer to list of options */
List DList ← List of decision objects DO;

Define Object Node = (DO, LoA for node, list of pointers corresponding to number of possible decisions);

// Method to build static tree of nodes recursively
Array Method D-Tree (n). // n = marker for current node
    If DList is not consumed
        Root ← DList[n];
        For (p ← 1 to number of decisions)
            create pointer;
            link pointer to DList (n+p);
            D-Tree (n+p). // build subtrees for each option
return;
```

This systemic methodology to structuring credential-based authentication-related factors allows for adding any emerging factors or deleting outdated ones from the structure.

6.7.3. The Credentials Light-Up Tree Module (CRLTM)

The CRLTM should implement the following tasks:

- Gathers information from messages passed between interacting entities for the current authentication instance to identify the involved LoA.Cr based factors.
- The retrieved data is used to flag the nodes of the credential tree signifying the Light-up tree nodes.

It should be noted that a universal identifier space, denoted in the database as END-ID, is setup with each END instance created to allow for clear communication between interacting parties such as END modules, CSPs, services and authentication servers. Although our evaluation doesn't include implementation of this module, the nodes of the credential tree corresponding to utilized credentials are directly flagged in the base tree structure for the purpose of evaluation.

6.7.4. The LoA Credentials Derivation Module (ALoAM)

The credentials derivation module (ALoAM) determines the aggregate value for credential-based LoA (or *ALoA.Cr)*. This module retrieves the LoA component values, relationship markers (additive, or low watermark) for the stored credentials. The derivation is accomplished using Algorithms presented in Section 6.6, and their performance analysis is presented in Section 6.7.

Once the *ALoA.Cr* is determined, it is stored in the LoA-DB along with interaction data including END-IDs for interacting entities and the time stamp of the interaction.

6.8. Chapter Summary

We previously proposed an architecture of a novel elastic, dynamic, and distributed authentication for M2C environment END AuthN based on three modes of authentication to establish trust in entities in the M2C environment. In this chapter we have investigated and analysed credential based LoA related factors. We introduced a methodology to structure these factors, then identified two types of relationships connecting those factors. We have also proposed alternative algorithms for combining those factors to determine an aggregate value for all credential based LoA related factors ALoA.Cr. In the next chapter, we will provide an evaluation of the END novel architecture proposed in chapters 5 and 6 of this work.

Chapter 7: Evaluation of END Authentication Architecture

7.1. Chapter Introduction

Chapters 5 and 6 presented the design and description of a novel architecture END for providing a standalone service for authentication in M2C. END AuthN service includes a trust framework (TruE) that considers three trust factors: LoA credential based, reputation based, and recommendation based. This chapter presents an evaluation of the proposed architecture. Section 7.2 describes the evaluation methodology. Sub-Sections 7.2.1 through 7.2.4 provides an evaluation of the work presented in this thesis as outlined in the evaluation methodology.

7.2. Evaluation Methodology

Since our research goal is to devise a secure and effective authentication solution, our evaluation of the work proposed in this thesis includes: An informal analysis against previously identified security threats of the MC-Model, an informal analysis of framework functionality against specified design requirements, a comparison with related work reviewed, and a performance evaluation of the proposed algorithms . The following sub-sections will cover these evaluation methods in order.

7.2.1. Informal Security Analysis

Entity authentication is one of the desired security requirements for devising an effective authentication solution for M2C to ensure that every interacting entity has the identity it claims. It should also guarantee that the authentication process continues throughout the resource utilization operation, starting from the registration step and including ensuring the authenticity of all requests and responses, whether the entities are part of the same domain or different domains. Non-repudiation should also be ensured for all interacting entities. Another important issue is interoperability, which allows heterogeneous systems to interact and collaborate securely. Although performance is not a security requirement per say, it is extremely vital that any security

solution does not hinder service provision by not incurring extra time or payload overhead and requiring minimal intervention.

Our analysis of the MC-model security in Chapter 3 enumerated the most anticipated threats for M2C. The security threats our work addresses are as follows:

1) Impersonation by a user of a malicious insider or outsider is one of the most common threats to cloud service provision. Some defence strategies against the impersonation of users include enforcing high composition rules for password choice and limiting the number of failed authentication attempts. Setting higher entropy requirements for secret tokens is another protection method. The use of cryptographic means to secure communication messages can also be enlisted for protection. Since our solution provides elastic authentication based on LoA and trust requirements, any of these measures can be enforced by CSPs by setting LoA requirements of 2 or higher for services that may suffer from password guessing attacks, phishing, pharming attacks, and session hijacking attacks.

2) Impersonation of a CSP by an attacker during communication between users and providers is a less common threat. This kind of attack can also be guarded against through the enforcement of higher level LoA requirements by CSPs, LoA 2 or higher, as well as securing communication messages by employing cryptographic means.

3) Alteration of authentication payload messages, including replaying, delaying, or modification of messages is another identified threat. Since our solution is an authentication as a service solution, it employs Communications channels for cloud computing which are usually protected by TLS.

7.2.2. Evaluation against Design Requirements

Following the design requirements specified in Section 5.6, this section focuses on how our solution can realize these requirements. Based on the MC-Model analysis presented in Chapter 3, we have identified several challenging issues and specified design requirements. Our novel solution has considered the suitable measures to fulfill these requirements as discussed below:

- R1: The first requirement was mutual authentication applying to all resource-access interactions the solution should allow for mutual authentication and should be continuous for all entities. The END AuthN architecture is designed as a stand-alone service able to receive and

handle authentication requests from any CSPs and services to authenticate clients and vice versa. So mutual authentication can be achieved if requested by interacting entities.

- R2: Another requirement was to support service access within different domains. Providing authentication-as-a-service (AaaS) for all entities in M2C also supports service access within different domains as well as service access within the same domain. When the END service is deployed by one entity, CSP1 for instance, it is initialized to gather authentication data based on credentials, reputation, and the recommendation of other entities interacting with CSP1. This will make it possible for other CSPs, services and clients from other domains to make use of the same instance of the END service.
- R3: A distinguishing requirement of our design is elasticity. Elasticity in cloud computing refers to the level of adaptability of computing solutions in response to fluctuating resource demand. However, we define elasticity in this work includes multi-level security provision- for authentication specifically- and performance variations to meet users' demands and needs. Security elasticity is achieved by allowing for different levels of assurance, multiple methods of authentication with various types of credentials, and a variable number of trusted entities. END design allows CSPs to decide the LoA requirements for the services they provide. END design can also adjust LoA through a variation of authentication Modes.

Performance elasticity is accomplished by allowing for a varying number of interacting entities, including users, services, and providers. The computational cost, number, and size of authentication messages can vary in accordance with entities' requirements and limitations.

The choice of parameters derived from the history of direct relations in the T(T.Rp) based on the nature of the interaction and the definition of SLA specification by CSPs and the services they provide.

The parameters used for T(T.Rc) determination are also chosen based on the nature of the interactions and the definition of SLA specification of CSPs and services they provide.

END design applies the idea of elasticity by allowing service providers to decide the LoA requirements for the services they provide. END design can also adjust LoA through a variation of authentication Modes.

- R4: Authentication solution should be scalable. Scalability means the ability of END service to adapt to increased demands in order to accommodate more CSPs, services, and clients. This

standalone cloud service will benefit from the scalability of the infrastructure, which is an underlying feature of CC. Hence, END service can grow or shrink with minimal user intervention.

7.2.3. Comparison to related work

We have presented a critical review of various Trust establishment models Based on the types of interactions considered for Trust calculations: direct, indirect or both. Our review also considered other criteria of trust, including **Weight assignment**, **False evaluations**, and **Centralization.** Table 7-1 summarizes characteristics of the methods we reviewed and includes our novel framework (TruE) for the purpose of comparison.

Table 7-1: Comparison of TruE with Related Work Reviewed

Model	Classification	Environment	Centralized	Consideration of Direct approach	Consideration of Indirect	Consideration of LoA	Consideration of False Evaluations	Weighted evaluations
TRAVOS [87]	Direct	Agent	×	✓	×	×	✓	✓
Geetha and Jayakumar [31]	Direct	Agent	×	✓	×	×	Tolerated if malice hosts < half hosts	×
Local Trust Model [25]	Direct	P2P	×	✓	×	×	✓	×
A recommendation based model for MANETs [78], [88]	Indirect	MANETs	×	×	✓	×	✓	×
A trust management model as SOA [85]	Hybrid	CC	×	✓	✓	×	_	×

FIRE [40]	Hybrid	Agent	×	✓	✓	×	×	×
Jurca and Faltings [45]	Hybrid	Agent	×	✓	✓	×	×	✓
Regret [75]	Hybrid	Agent	×	✓	✓	×	✓	×
Basheer et al. [12]	Hybrid	Agent	×	✓	✓	×	×	×
Model based on Service Parametric Model [79]	Hybrid	CC	✓	✓	✓	×	_	✓
Yao & Zhang [103]	LoA based	Grid	✓	×	×	✓	_	✓
Our novel framework (TruE)	Hybrid	M2C	✓	✓	✓	✓	✓	✓

Upon examination of the comparison summarized in Table 7-1 and the detailed review presented in Chapter 5, we make the following observations:

- Most Trust Models for CC focus on the evaluation of the Quality of Service. Our novel TruE framework, however, allows for the definition of various trust metrics as per service providers' requirements.
- Another observation we make is that the reviewed Trust Models for CC examine Trust Levels for Cloud Services and Cloud Service Providers to guide potential users in choosing the most suitable services offered. However, our scheme considers evaluations of users as well in order to establish trust for the purpose of authenticating users requesting services.
- We also note that recommendation-based trust techniques, or indirect models, aim to discover misbehaving entities before interaction in order to avoid potential bad experiences. Although this is more commonly considered for models employed in agent environments or P2P environments, the novel TruE framework accomplishes the same goal. Since our framework is the basis for authenticating entities, misbehaving entities would not accumulate enough trust to be aggregated for the sake of a secure authentication decision.

- It is also notable that all the trust models we reviewed for various computing environments don't consider credentials when quantifying trust for the entity in question. The exception is the work presented in [103] which bases trust quantification on credentials for the purpose of authenticating entities in a grid computing environment. Our novel trust framework also considers credentials offered at the time of authentication as a trust factor for M2C computing environments. Our work also includes system to system authentication in addition to, user to system authentication.
- Our novel work uniquely offers quantification of trust based on the three modes of authentication as described in Section 5.5. All other reviewed trust models in Table 7-1 are either reputation-based, recommendation based, reputation and recommendation based, or credential based. However, we aimed in our work to authenticate entities in M2C based on all authentication resources that may be available at time of authentication including reputation-based, recommendation-based, and credential-based factors.
- Furthermore, more recent research reviewed also lacks consideration of all three modes for establishing trust such as the research presented in [98], [92], [53], [9], [43], [64], [67].

The trust framework we proposed is the basis of the novel END AuthN architecture. Accordingly, we provided an analytical review of authentication solutions in distributed systems to identify areas for improvement (Chapter 4). Table 7-2 summarizes features of the methods we reviewed and compared to those of our novel architecture (END).

Table 7-2: Comparison of END with Related Work Reviewed

		Mutual Authentication	Cross domain	Continuous
P2P	**Enc PWD [11]**	✖	✖	✖
	PKI based [44], [54]	✓	✖	✖
	IBC [54], [65]	✓	✓	✖
	CL-PKC [54]	✓	✓	✖

GC	**Static PWD [29], [39]**	✖	✓	✖
	Dynamic PWD [29], [39]	✖	✓	✖
	Kerberos [83], [29], [52]	✓	✓	✖
	PKI based [29], [39], [14]	✓	✓	✖
	SPTC [56]	✓	✓	✖
	IBC [29], [41], [35]	✓	✓	✖
	CL-PKC [29], [30]	✓	✓	✖
	Pairing-free CL-PCK [30]	✓	✓	✖
CC	**Multi-level PWD [26]**	✖	✓	✖
	TCG [81]	✖	✓	✖
	CAM [13]	✓	✓	✖
	Dynamic Multi-factor [10]	✖	✓	✖
	InterCloud [17], [57]	✖	✓	✖
	ECC ID-based [22]	✓	✓	✖
M2C	**Novel authentication solution END**	✓	✓	✓

Upon examination of the comparison, presented in Table 7-2 and the detailed review presented in Chapter 4, we observe the following:

- All authentication methods for P2P systems do not meet most of the requirements incurred by the M2C features because P2P is based on equal weight anonymous nodes (peers) collaborating on a given task. Accordingly, the goal and requirements of P2P authentication are different and not suited to those related to M2C environment.
- As for authentication solutions for Grid Computing, Kerberos is the scheme that covers most requirements. However, it doesn't fulfil the requirements of continuous authentication.
- The requirement of continuous authentication is not met by any of the reviewed methods. The novel authentication architecture presented in this thesis (END) can provide continuous authentication as per the workflow of the task at hand. When the END service is employed, it allows entities to register, login to one CSP or service and continues to provide authentication when services requesting other services on behalf of the user.
- Interoperability is a key requirement that is not achieved by most of the examined methods. On the other hand, the END standalone service provides a layer for authentication provision, allowing different CSPs to collaborate based on standardized communication protocols.
- Additionally, more recent work also doesn't meet all three requirements highlighted in Table 7-2 such as the work presented in [69], [7].

Hence, the novel authentication architecture END provides fine-grained authentication for the M2C model.

7.2.4. Performance Evaluation

In this section we conduct performance evaluation for the various algorithms introduced in this thesis. Section 7.2.4.1 will investigate the performance of algorithms proposed for performing Low watermark rule of combining dependent LoA-related factors. Section 7.2.4.2 investigates performance of algorithms proposed for performing the additive rule of combining independent LoA-related factors. In each case, evaluation experiments are carried out to measure computational times consumed by the algorithm examined. The experiments are hosted on a 2.9 GHz Intel Core i7 Mac Book Pro with 16 GB 1600 MHz DDR3 memory and Mac OS High Sierra ver. 10.13.6. Algorithms are implemented as a Java application with Java SETM Runtime Environment version 8u31-macosx-x64. Performance monitoring tool was YourKit Java Profiler 2018.04-b83, which provides regression analysis among other measuring tools for CPU and memory usage analysis.

7.2.4.1. Performance Evaluation of Low Watermark Algorithms

For a number of dependent LoA-related credential-based factors, we examined two algorithms: the sorting algorithm from Java API enhancements in Java SE 7, and the pairwise comparisons presented in Section 6.6.1. We predicted that the pairwise comparison algorithm will grow slower than the sorting algorithm as the input set (number of factors) grows. This is due to the fact that the computational complexity of the sorting algorithm is O(n × log(n)), hence computation time would grow in Loglinear time as the input set (number of factors) becomes larger. On the other hand, the computational complexity of the pairwise comparison algorithm is O(n), which is linear complexity.

We run an initial experiment to determine the number of iterations (n) necessary to run the performance analysis experiment. Both algorithms are run at an incremental number of iterations to limit the arbitrary effect on the execution time. A regression analysis is performed, showing that n = 50000 is adequate for the sort algorithm (Figure. 7-1) and n = 40000 for the pairwise comparison (Figure. 7-2).

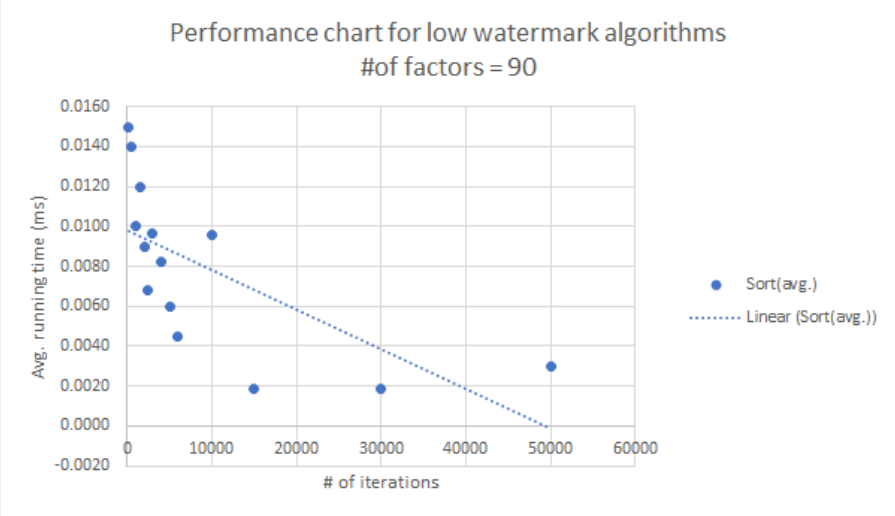

Figure 7-1. Linear Regression for the Sort Algorithm Plot

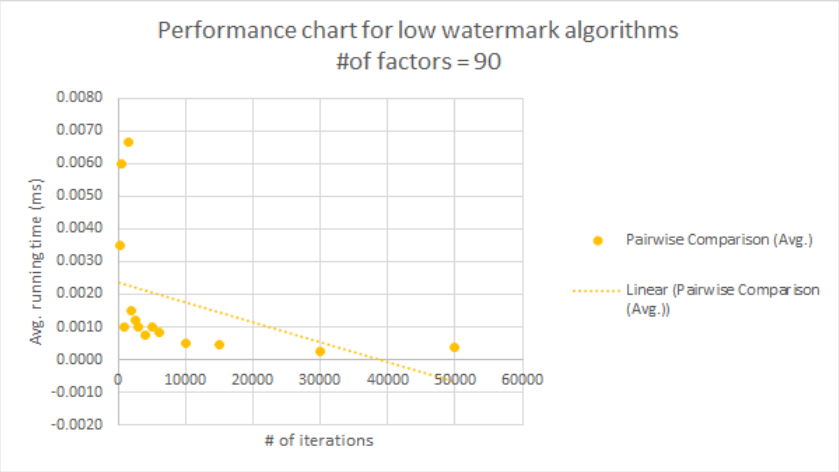

Figure 7-2. Linear Regression for the Pairwise Comparison Algorithm Plot

Hence, we run the performance evaluation comparison with n = 50000. The results are illustrated in Figure 7-2. As the figure below depicts, the pairwise comparison proves to be more efficient with regard to execution time. It also grows at a slower rate than the sort algorithm, as the number of factors grows larger, according to the anticipated results.

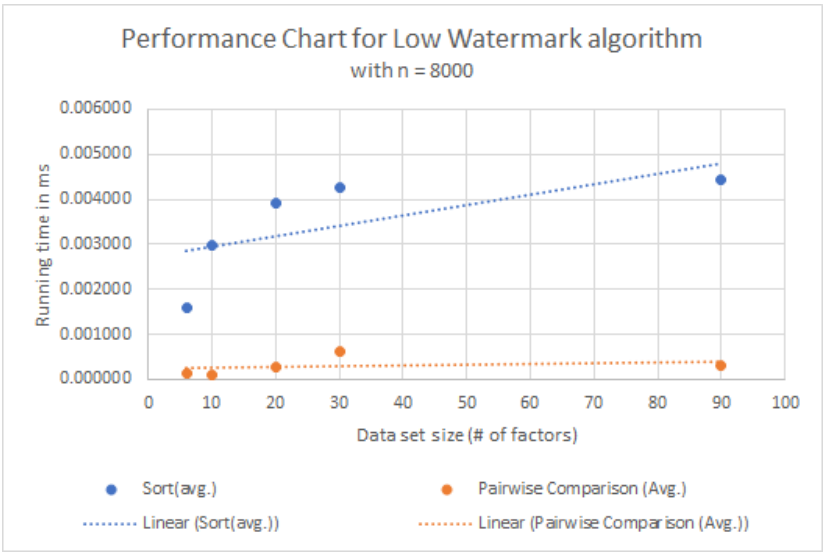

Figure 7-3. Comparison of Performance evaluation between Sort algorithm and Pairwise comparison algorithm

From the performance evaluation conducted in this section, we decide to utilize the pairwise algorithm for the evaluation of our proposed architecture.

7.2.4.2. Performance Evaluation of Additive Algorithms

For a set of independent LoA-related credential-based factors, we examined three algorithms: $\mathrm{P.LoA_{Add}}$, $\mathrm{M.LoA_{Add}}$, and $\mathrm{C.LoA_{Add}}$ as presented in Section 6.6.2. We predict that all three algorithms will grow in a similar fashion as the size of the input grows. This is due to the fact that the computational complexity of all additive algorithms is O(n), which is linear time.

We run an initial experiment to determine the number of iterations (n) necessary to run the performance analysis experiment. A regression analysis of all three algorithms yielded the following result for adequate n values: n > 350000 for the $\mathrm{P.LoA_{Add}}$ algorithm, n = 300000 for the $\mathrm{M.LoA_{Add}}$ algorithm, and n = 275000. Hence, the performance evaluation was run for n= 350000 iterations. The comparative graph of the three algorithms is shown in Figure 7-4. It shows, as predicted, similar behavior for all examined algorithms. The $\mathrm{C.LoA_{Add}}$ algorithm shows a slight advantage over the other two algorithms as the input size grows.

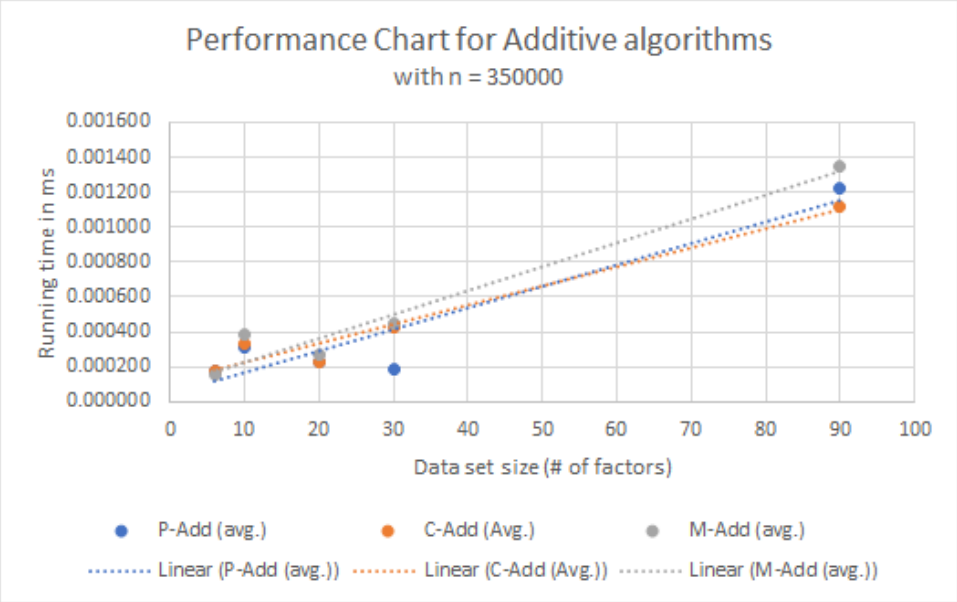

Figure 7-4. Comparative Analysis of Computational Time of three Additive Rule Algorithms

Based on the performance evaluation conducted in this section, we decide to utilize the $\mathrm{C.LoA_{Add}}$ algorithm for the evaluation of our proposed architecture.

7.3. Chapter Summary

This chapter has described an evaluation methodology for the END authentication solution. END AuthN service includes a trust framework (TruE) that bases authentication decisions on the evaluation of aggregated trust values for M2C entities. Firstly, the proposed solution was evaluated against the design requirements specified earlier in the thesis and found to be in line with desired requirements. Then, the work presented in the thesis was evaluated with regard to security threats identified previously in our work and showed how the solution safeguards against these threats. We also provided a comparison of our novel trust framework (TruE) and trust models reviewed, as well as a comparison of our novel authentication architecture (END) with authentication solutions from various distributed computing paradigms. The comparison shows that the novel END authentication fulfils the goal we set out to accomplish and provides a novel methodology for authentication in the M2C environment by employing the concept of elasticity. Finally, we provided a performance evaluation of several algorithms proposed throughout the description of the novel END authentication.

The next chapter concludes this thesis and gives recommendations for future work.

Chapter 8: Conclusion and Future Work

The aim of this thesis is to propose and design an authentication solution for Multi Cloud Computing. This chapter provides a summary of all the research done and the contributions of the thesis and gives recommendations for future work.

8.1. Thesis Summary

The work in this thesis is structured as follows:

- Research background,
- Problem identification and MC Model derivation,
- Literature review of authentication solutions and trust establishment in distributed computing environments,
- Proposal and design of TruE trust framework and END authentication architecture,
- Evaluation and conclusion of the work presented.

Research Background

Chapter 2 of this thesis has presented background research on Cloud Computing (CC) and Multiple Cloud Computing (M2C). It included an overview of CC, its general characteristics, and deployment and service models. It also introduced the security issues and challenges in this environment. Chapter 4 provided background information on the authentication process, types of tokens, and threats against authentication.

Literature Review

Chapter 3 concentrated on the literature review of authentication solutions for Distributed Systems including Peer-to-Peer systems, Grid Computing, and Cloud Computing and provided an analytical comparison of these solutions to identify the knowledge gap in authentication for M2C and high-level ideas for our novel solution. Chapter 5 reviewed trust models that may be employed in distributed systems such as P2P, Grid, and Cloud Computing in order to identify a novel framework for establishing trust in M2C.

Problem Identification and MC-Model derivation

Chapter 4 presented and analyzed a real-life use case of M2C to provide insight into the context and security provisions within M2C. This was accomplished by describing three solutions to a real-life use case in order to describe the MC-Model, a generic model for M2C. We conducted a threat analysis of MC-Model to derive security requirements for M2C, with a focus on authentication related requirements. It also presented high-level ideas for our novel solution to authentication provision for the M2C.

Design of TruE trust framework and END AuthN service

Chapter 5 introduced the novel framework TruE for establishing trust for interacting entities in M2C. It also provided a general description of the novel proposed architecture (END AuthN) and a detailed description of its architectural components. Chapter 6 included an analysis of credential based LoA-related factors and defined relationships between them. It also proposed a novel methodology for structuring these factors in order to combine them to compute an aggregate value of trust based on these factors.

Evaluation and Conclusion

Chapter 7 provides an evaluation of the proposed architecture against design requirements, and against security threats identified for M2C. Further, Chapter 8 summarizes the research done, the contributions of the thesis, and recommendations for future work.

8.2. Contributions

The contributions of this thesis are summarized below:

• The description of a novel generic abstract model for multi-cloud solutions (**MC-Model**) is presented in chapter 3. The MC-Model was based on the description of three incremental solutions of a real-life use case.

• **Threat Analysis of Multi Cloud Computing** (M2C) based on the MC-Model presented in Chapter 3. It provided an investigation of the potential threats leading to security breaches by outsiders or any of the entities within the model.

• **Methodology for structuring LoA related factors** in Multi Cloud Computing (M2C) is presented in Chapter 6. We proposed a Decision Tree to build a tree structure connecting LoA-related factors based on authentication processes and tokens used in a given authentication instance.

• Algorithm for calculating aggregate LoA related factors based on credentials ($\boldsymbol{ALoA.Cr}$) for Multi Cloud Computing (M2C) entities for a given authentication instance. This is described in chapter 6.

• Chapter 6 also proposes three algorithms namely, the Weighted Average Algorithm ($\mathbf{W.LoA_{Add}}$), the Mapped Weight Algorithm ($\mathrm{M.LoA_{Add}}$), and the Combined LoA Algorithm ($\mathrm{C.LoA_{Add}}$), for evaluating an aggregate value of LoA for independent credential based LoA related factors.

• Introduction to the concept of **three modes of authentication** (based on credentials, based on interaction history, and based on recommendations from other entities).

• Introduction to the concept of **Authentication-as-a-Service (A3S)**. It provides a distributed and dynamic on-demand authentication solution.

• A novel framework for establishing trust in M2C entities (**Trust Engine or TruE**). It considers three trust establishing factors, including credentials provided by the claimant at the time of authentication, reputation of varied cloud entities (providers, services, and clients) based on their direct interactions, and recommendations of cloud entities based on collected entity evaluations of previous indirect interactions.

• A novel elastic, dynamic, and distributed authentication solution **END AuthN** for M2C based on trust establishment provided by (TruE) framework.

8.3. Future Work

We give the following recommendations as directions for future work:

•The work in this thesis assumes that the interactions between M2C entities can be captured from the control messages exchanged between them. However, a methodology for capturing the indicated data was not offered in our work. Proposal of such a methodology is recommended to enhance the proposed work.

- Another future direction of the work presented in this thesis would provide a definition of a specialized protocol to govern communication necessary for the proposed idea to work more effectively and efficiently.
- Evaluation of our work performance was limited to the assessment of single algorithms. A fully functioning prototype of the proposed work would provide a more effective methodology for evaluating the performance of the architecture against current authentication solutions for M2C.
- Another alternative to better evaluation can be accomplished through a set of experiments with real-life use-case scenarios such as the one described to derive the MC-Model. The TruE framework relies on the evaluation of trust based on the reputation and recommendation of entities. We suggested a few simplistic metrics to measure the behavior of entities. The definition of more metrics can be further investigated and evaluated.
- The weights of the trust factors we consider in our solution are either static in nature or semi-static with values initialized as per service provider's definition. More complex mathematical methods such as multi-criteria decision-making methodologies can be further researched to evaluate their effectiveness as part of our trust framework.
- The Authentication service can be finetuned by providing alternative scenarios to rejected clients, to match their trust level. So, if a cloud client has trust values that are insufficient for given cloud services, the service can negotiate alternative services to accomplish the same task.

Bibliography

[1] "ITU-Tfg Cloud TR," Sector, Standarization

ITU, OF, 2012.

[2] J. Abawajy, "Determining service trustworthiness in intercloud computing environments," in *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 2009: IEEE, pp. 784-788.

[3] I. Agudo, C. Fernandez-Gago, and J. Lopez, "A model for trust metrics analysis," in *International Conference on Trust, Privacy and Security in Digital Business*, 2008: Springer, pp. 28-37.

[4] U. Ahmed, I. Raza, and S. A. Hussain, "Trust evaluation in cross-cloud federation: Survey and requirement analysis," *ACM Computing Surveys (CSUR),* vol. 52, no. 1, pp. 1-37, 2019.

[5] S. I. Akhtar, A. Rauf, H. Abbas, and M. F. Amjad, "Inter Cloud Interoperability Use Cases and Gaps in Corresponding Standards," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2020: IEEE, pp. 585-592.

[6] M. Alaluna, L. Ferrolho, J. R. Figueira, N. Neves, and F. M. Ramos, "Secure multi-cloud virtual network embedding," *Computer Communications,* vol. 155, pp. 252-265, 2020.

[7] T. Alyas *et al.*, "Multi-Cloud integration security framework using honeypots," *Mobile Information Systems,* vol. 2022, pp. 1-13, 2022.

[8] A. Amazon, "Amazon Web Services Overview of Security Processes," Amazon, AWS, 2015.

[9] K. A. Awan, I. U. Din, A. Almogren, and J. J. Rodrigues, "AutoTrust: A privacy-enhanced trust-based intrusion detection approach for internet of smart things," *Future Generation Computer Systems,* vol. 137, pp. 288-301, 2022.

[10] R. K. Banyal, P. Jain, and V. K. Jain, "Multi-factor authentication framework for cloud computing," in *2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation*, 2013: IEEE, pp. 105-110.

[11] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," *arXiv preprint cs/0412017,* 2004.

[12] G. S. Basheer, M. S. Ahmad, A. Y. Tang, and S. Graf, "Certainty, trust and evidence: Towards an integrative model of confidence in multi-agent systems," *Computers in Human Behavior,* vol. 45, pp. 307-315, 2015.

[13] K. Beaty, A. Kundu, V. Naik, and A. Acharya, "Network-level access control management for the cloud," in *2013 IEEE International Conference on Cloud Engineering (IC2E)*, 2013: IEEE, pp. 98-107.

[14] A. Bendahmane, M. Essaaidi, A. El Moussaoui, and A. Younes, "Grid computing security mechanisms: State-of-the-art," in *2009 International Conference on Multimedia Computing and Systems*, 2009: IEEE, pp. 535-540.

[15] S. Buchegger and J.-Y. Le Boudec, "A robust reputation system for peer-to-peer and mobile ad-hoc networks," in *P2PEcon 2004*, 2004, no. LCA-CONF-2004-009.

[16] W. Burr *et al.*, "NIST Special Publication 800-63-2 Electronic Authentication Guideline," *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology,* 2013.

[17] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Security and cloud computing: Intercloud identity management infrastructure," in *2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2010: IEEE, pp. 263-265.

[18] S. Chasins *et al.*, "The sky above the clouds," *arXiv preprint arXiv:2205.07147,* 2022.

[19] V. Chauhan and A. Singh, "Security Pitfalls in Multi-Cloud Computing Environment," *M. Tech Student Chandigarh university, Assistant Professor Indian Institute of Technology Ropar, Department of Computer Science and Engineering,* 2009.

[20] C. Chen, H. Lin, and C. Gun, "A fair and dynamic password authentication system," in *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, 2011: IEEE, pp. 4505-4509.

[21] M. Chen, K. Wu, C. Wu, and Z. Wu, "Certificateless-Signature-Based Authenticated Key Agreement Protocol for Grid," in *2010 Fifth Annual ChinaGrid Conference*, 2010: IEEE, pp. 267-270.

[22] T.-H. Chen, H.-l. Yeh, and W.-K. Shih, "An advanced ecc dynamic id-based remote mutual authentication scheme for cloud computing," in *2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering*, 2011: IEEE, pp. 155-159.

[23] J. Chin, N. Zhang, A. Nenadic, and O. Bamasak, "A context-constrained authorisation (cocoa) framework for pervasive grid computing," *Wireless Networks,* vol. 16, no. 6, pp. 1541-1556, 2010.

[24] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006, pp. 171-178.

[25] Y.-x. Cui, H.-y. Ning, and Y.-y. Du, "A local trust model based on time influence function in P2P networks," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2015: IEEE, pp. 507-510.

[26] H. Dinesha and V. K. Agrawal, "Multi-level authentication technique for accessing cloud services," in *2012 International Conference on Computing, Communication and Applications*, 2012: IEEE, pp. 1-4.

[27] U. Divakarla and K. Chandrasekaran, "Trusted path between two entities in Cloud," in *2016 6th International Conference-Cloud System and Big Data Engineering (Confluence)*, 2016: IEEE, pp. 157-162.

[28] Y. Elkhatib, "Mapping cross-cloud systems: Challenges and opportunities," in *8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[29] A. Farouk, A. A. Abdelhafez, and M. M. Fouad, "Authentication mechanisms in grid computing environment: Comparative study," in *2012 International Conference on Engineering and Technology (ICET)*, 2012: IEEE, pp. 1-6.

[30] A. Farouk, A. Miri, M. M. Fouad, and A. A. Abdelhafez, "Efficient pairing-free, certificateless two-party authenticated key agreement protocol for grid computing," in *2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, 2014: IEEE, pp. 279-284.

[31] G. Geetha and C. Jayakumar, "Implementation of trust and reputation management for free-roaming mobile agent security," *IEEE Systems Journal,* vol. 9, no. 2, pp. 556-566, 2014.

[32] L. Goasduff. "Why Organizations Choose a Multicloud Strategy." https://www.gartner.com/smarterwithgartner/why-organizations-choose-a-multicloud-strategy/ (accessed.

[33] L. Gu and M. A. Gregory, "A green and secure authentication for the 4th generation mobile network," in *2011 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, 2011: IEEE, pp. 1-7.

[34] S. M. Habib, S. Hauke, S. Ries, and M. Mühlhäuser, "Trust as a facilitator in cloud computing: a survey," *Journal of Cloud Computing: Advances, Systems and Applications,* vol. 1, no. 1, pp. 1-18, 2012.
[35] M. Hedayati, S. H. Kamali, and R. Shakerian, "Using identity-based secret public keys cryptography for heuristic security analyses in grid computing," in *2010 5th International Symposium on Telecommunications*, 2010: IEEE, pp. 221-230.
[36] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *10th international conference on autonomic computing ({ICAC} 13)*, 2013, pp. 23-27.
[37] J. Hong, T. Dreibholz, J. A. Schenkel, and J. A. Hu, "An overview of multi-cloud computing," in *Workshops of the International Conference on Advanced Information Networking and Applications*, 2019: Springer, pp. 1055-1068.
[38] J. Huang and D. M. Nicol, "Trust mechanisms for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications,* vol. 2, pp. 1-14, 2013.
[39] W. Huang, W. Du, L. Guo, and L. Huang, "Electronic trading systems in several authentication methods commonly used comparison," in *2010 WASE International Conference on Information Engineering*, 2010, vol. 2: IEEE, pp. 251-254.
[40] T. D. Huynh, N. R. Jennings, and N. Shadbolt, "FIRE: An integrated trust and reputation model for open multi-agent systems," in *6th European Conference on Artificial Intelligence, Valencia, Spain, 2004-01-01*, 2004.
[41] S. A. Ismail and M. A. Ngadi, "New security authentication mechanisms in grid computing web environment," in *2011 International Conference on Research and Innovation in Information Systems*, 2011: IEEE, pp. 1-4.
[42] W. Jansen and T. Grance, "Sp 800-144. guidelines on security and privacy in public cloud computing," ed: National Institute of Standards & Technology, 2011.
[43] J. Jiang, H. Wang, and W. Li, "A Trust model based on a time decay factor for use in social networks," *Computers & Electrical Engineering,* vol. 85, p. 106706, 2020.
[44] W. K. Josephson, E. G. Sirer, and F. B. Schneider, "Peer-to-peer authentication with a distributed single sign-on service," in *International Workshop on Peer-to-Peer Systems*, 2004: Springer, pp. 250-258.
[45] R. Jurca and B. Faltings, "Enforcing truthful strategies in incentive compatible reputation mechanisms," in *International Workshop on Internet and Network Economics*, 2005: Springer, pp. 268-277.
[46] J. Kemp *et al.*, "Authentication Context for the OASIS Security Assertion Markup Language," 2005.
[47] I. M. Khalil, A. Khreishah, and M. Azeem, "Cloud computing security: A survey," *Computers,* vol. 3, no. 1, pp. 1-35, 2014.
[48] H. H. Kilinc and T. Yanik, "A survey of SIP authentication and key agreement schemes," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 2, pp. 1005-1023, 2013.
[49] D.-S. Kim and K.-S. Hong, "Multimodal biometric authentication using teeth image and voice in mobile environment," *IEEE Transactions on Consumer Electronics,* vol. 54, no. 4, pp. 1790-1797, 2008.
[50] A. Kounoudes, V. Kekatos, and S. Mavromoustakos, "Voice biometric authentication for enhancing Internet service security," in *2006 2nd International Conference on Information & Communication Technologies*, 2006, vol. 1: IEEE, pp. 1020-1025.
[51] M. Lacoste *et al.*, "User-centric security and dependability in the clouds-of-clouds," *IEEE Cloud Computing,* vol. 3, no. 5, pp. 64-75, 2016.

[52] I. Lahmer and N. Zhang, "MapReduce: MR model abstraction for future security study," in *Proceedings of the 7th International Conference on Security of Information and Networks*, 2014, pp. 392-398.
[53] H. Leng, Y. Zhao, and D. Wang, "Message passing approach for social contagions based on the trust probability with multiple influence factors," *Physica A: Statistical Mechanics and its Applications,* vol. 587, p. 126510, 2022.
[54] Z. Li, X. Xu, L. Shi, J. Liu, and C. Liang, "Authentication in peer-to-peer network: Survey and research directions," in *2009 Third International Conference on Network and System Security*, 2009: IEEE, pp. 115-122.
[55] F. Liu *et al.*, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications,* vol. 20, no. 3, pp. 14-22, 2013.
[56] G. Liu, "Using Security Proxy Based Trusted Computing Enhanced Grid Security Infrastructure," in *2010 2nd International Conference on Information Engineering and Computer Science*, 2010: IEEE, pp. 1-4.
[57] R. K. Lomotey and R. Deters, "Saas authentication middleware for mobile consumers of iaas cloud," in *2013 IEEE Ninth World Congress on Services*, 2013: IEEE, pp. 448-455.
[58] P. Lozhnikov and O. Chernikova, "Handwriting dynamics as a means of authentication," in *2011 International Conference for Internet Technology and Secured Transactions*, 2011: IEEE, pp. 176-179.
[59] T. Luxner, "Cloud Computing Trends: 2021 State of the Cloud Report," ed, 2021.
[60] D. C. Marinescu, *Cloud computing: theory and practice*. Morgan Kaufmann, 2022.
[61] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.
[62] P. Mishra *et al.*, "Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2. 0," ed: OASIS SSTC, March, 2005.
[63] M. B. Monir, M. H. AbdelAziz, A. A. AbdelHamid, and E.-S. M. EI-Horbaty, "Trust management in cloud computing: a survey," in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2015: IEEE, pp. 231-242.
[64] N. Narang and S. Kar, "A hybrid trust management framework for a multi-service social IoT network," *Computer Communications,* vol. 171, pp. 61-79, 2021.
[65] K. V. Nguyen, "Simplifying peer-to-peer device authentication using identity-based cryptography," in *International conference on Networking and Services (ICNS'06)*, 2006: IEEE, pp. 43-43.
[66] Z. Noorian and M. Ulieru, "The state of the art in trust and reputation systems: a framework for comparison," *Journal of theoretical and applied electronic commerce research,* vol. 5, no. 2, pp. 97-117, 2010.
[67] P. Parmar and M. Bhavsar, "Achieving Trust using RoT in IaaS Cloud," *Procedia Computer Science,* vol. 167, pp. 487-495, 2020.
[68] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *2009 Second international symposium on information science and engineering*, 2009: IEEE, pp. 23-27.
[69] D. Prabakaran and S. Ramachandran, "Multi-factor authentication for secured financial transactions in cloud environment," *CMC-Computers, Materials & Continua,* vol. 70, no. 1, pp. 1781-1798, 2022.
[70] M. Purkhiabani and A. Salahi, "Enhanced authentication and key agreement procedure of next generation evolved mobile networks," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, 2011: IEEE, pp. 557-563.

[71] C. Ramalingam and P. Mohan, "Addressing Semantics Standards for Cloud Portability and Interoperability in Multi Cloud Environment," *Symmetry,* vol. 13, no. 2, p. 317, 2021. [Online]. Available: https://www.mdpi.com/2073-8994/13/2/317.

[72] X. Ren and X.-W. Wu, "A novel dynamic user authentication scheme," in *2012 International Symposium on Communications and Information Technologies (ISCIT)*, 2012: IEEE, pp. 713-717.

[73] F. Richter, "Amazon Leads $150-Billion Cloud Market," July 5th, 2021 2021. [Online]. Available: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/.

[74] M. D. Ryan, "Cloud computing privacy concerns on our doorstep," *Communications of the ACM,* vol. 54, no. 1, pp. 36-38, 2011.

[75] J. Sabater, "Evaluating the ReGreT system," *Applied Artificial Intelligence,* vol. 18, no. 9-10, pp. 797-813, 2004.

[76] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, 2002, pp. 475-482.

[77] A. S. A. Saleh, E. M. R. Hamed, and M. Hashem, "Building trust management model for cloud computing," in *2014 9th International Conference on Informatics and Systems*, 2014: IEEE, pp. PDC-116-PDC-125.

[78] A. M. Shabut, K. P. Dahal, S. K. Bista, and I. U. Awan, "Recommendation based trust model with an effective defence scheme for MANETs," *IEEE Transactions on Mobile Computing,* vol. 14, no. 10, pp. 2101-2115, 2015.

[79] S. Sharma and M. Mahrishi, "Implementation of trust model on CloudSim based on service parametric model," in *2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2015: IEEE, pp. 351-356.

[80] D. Shehada, M. J. Zemerly, C. Y. Yeun, M. Al Qutayri, and Y. Al Hammadi, "A framework for comparison of trust models for multi agent systems," in *2015 International Conference on Information and Communication Technology Research (ICTRC)*, 2015: IEEE, pp. 318-321.

[81] Z. Shen and Q. Tong, "The security of cloud computing system enabled by trusted computing technology," in *2010 2nd International Conference on Signal Processing Systems*, 2010, vol. 2: IEEE, pp. V2-11-V2-15.

[82] S. Singh and M. Yamini, "Voice based login authentication for Linux," in *2013 International Conference on Recent Trends in Information Technology (ICRTIT)*, 2013: IEEE, pp. 619-624.

[83] W. Stallings, *Cryptography and network security, 5th Edition*. New York: Practice HAll, 2011.

[84] Y. Sui, X. Zou, E. Y. Du, and F. Li, "Design and analysis of a highly user-friendly, secure, privacy-preserving, and revocable authentication method," *IEEE Transactions on Computers,* vol. 63, no. 4, pp. 902-916, 2013.

[85] W. Tan, Y. Sun, L. X. Li, G. Lu, and T. Wang, "A Trust Service-Oriented Scheduling Model for Workflow Applications in Cloud Computing," *IEEE Systems Journal,* vol. 8, no. 3, pp. 868-878, 2013.

[86] M. Tavis and P. Fitzsimons, "Web application hosting in the aws cloud: Best practices," *Relatório técnico, Amazon, Setembro,* vol. 11, pp. 12-15, 2012.

[87] W. L. Teacy, J. Patel, N. R. Jennings, and M. Luck, "Travos: Trust and reputation in the context of inaccurate information sources," *Autonomous Agents and Multi-Agent Systems,* vol. 12, pp. 183-198, 2006.

[88] G. Theodorakopoulos and J. S. Baras, "Trust evaluation in ad-hoc networks," in *Proceedings of the 3rd ACM workshop on Wireless security*, 2004, pp. 1-10.

[89] D. Todorov, *Mechanics of user identification and authentication: Fundamentals of identity management*. CRC Press, 2007.

[90] B. K. Tripathy, P. Bera, and M. A. Rahman, "Analysis of trust models in Mobile Ad Hoc Networks: A simulation based study," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, 2016: IEEE, pp. 1-8.
[91] M. Trojahn and F. Ortmeier, "Toward mobile authentication with keystroke dynamics on mobile phones and tablets," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, 2013: IEEE, pp. 697-702.
[92] J. M. J. Valero, P. M. S. Sánchez, M. G. Pérez, A. H. Celdrán, and G. M. Pérez, "Toward pre-standardization of reputation-based trust models beyond 5G," *Computer Standards & Interfaces,* vol. 81, p. 103596, 2022.
[93] J. Varia and S. Mathew, "Overview of amazon web services," *Amazon Web Services,* vol. 105, 2014.
[94] J. Wan, Z. Liu, K. Zhou, and R. Lu, "Mobile cloud computing: application scenarios and service models," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2013.
[95] Z. Wei, H. Tang, F. R. Yu, M. Wang, and P. Mason, "Security enhancements for mobile ad hoc networks with trust management using uncertain reasoning," *IEEE Transactions on Vehicular Technology,* vol. 63, no. 9, pp. 4647-4658, 2014.
[96] M. Werlinder and E. Tham, "Application of Amazon Web Services in software development."
[97] X. Xiaoping and Y. Junhu, "Research on cloud computing security platform," in *2012 Fourth International Conference on Computational and Information Sciences*, 2012: IEEE, pp. 799-802.
[98] Y. Xu, Z. Gong, J. Y.-L. Forrest, and E. Herrera-Viedma, "Trust propagation and trust network evaluation in social networks based on uncertainty theory," *Knowledge-Based Systems,* vol. 234, p. 107610, 2021.
[99] Y. H. Yahaya, M. R. M. Isa, and M. I. Aziz, "Fingerprint biometrics authentication on smart card," in *2009 Second International Conference on Computer and Electrical Engineering*, 2009, vol. 2: IEEE, pp. 671-673.
[100] R. Yaich, N. Cuppens, and F. Cuppens, "Enabling Trust Assessment In Clouds-of-Clouds: A Similarity-Based Approach," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017, pp. 1-9.
[101] L. Yao, *A structured approach to electronic authentication assurance level derivation*. The University of Manchester (United Kingdom), 2010.
[102] L. Yao and N. Zhang, "A Generic Authentication LoA Derivation Model," in *Emerging Challenges for Security, Privacy and Trust: 24th IFIP TC 11 International Information Security Conference, SEC 2009, Pafos, Cyprus, May 18–20, 2009. Proceedings 24*, 2009: Springer, pp. 98-108.
[103] L. Yao and N. Zhang, "Quantifying authentication levels of assurance in grid environments," in *2010 Sixth International Conference on Information Assurance and Security*, 2010: IEEE, pp. 298-303.
[104] W. Zeng, J. Zhao, and M. Liu, "Several public commercial clouds and open source cloud computing software," in *2012 7th International Conference on Computer Science & Education (ICCSE)*, 2012: IEEE, pp. 1130-1133.
[105] P. Zhang, A. Durresi, and L. Barolli, "Survey of trust management on various networks," in *2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, 2011: IEEE, pp. 219-226.
[106] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation computer systems,* vol. 28, no. 3, pp. 583-592, 2012.

Appendix A

NIST E-Authentication Level of Assurance

NIST E-Authentication Guidelines describe four levels of assurance [16]: Level 1, Level 2, Level 3, and Level 4. These LoA-related factors are classified into several categories, defined next [16].

2. <u>Registration and Identity Proofing (R and IdP)</u>

Registration is a process of proceeding with authentication, where an applicant applies to become a Subscriber of a Credential Provider (CP) and receives identity credentials.

Identity Proofing is a process by which a CP and a Registration Authority (RA) collect and verify information about a person for the purpose of issuing credentials to that person.

The R and IdP process should ensure that [16]:

- An entity with the applicant's attributes exists, and those attributes are sufficient to uniquely identify a single entity.
- The Applicant whose token is registered is, in fact, the entity entitled to the identity.
- It is difficult for an entity to repudiate the registration and dispute an authentication using this token later. Table A1 shows LoA values for varying R and IdP processes.

Table A1. LoA for Registration and ID Proofing [16]

LoA value	R and IP Process
1	No specific requirements. Some effort should be made to uniquely identify applicants.
2	• Applicant supplies a valid current government ID such as a driver's license. • RA inspects for the correct format of information input and verifies it with the applicable agency. • Address/phone number confirmation and notification are as follows: 1. CP issues credentials to confirm the ability of the applicant to receive e-mail or telephone communications (call or text message). Any secret sent over an unprotected session will be reset upon first use. 2. RA or CP sends notice to the address confirmed in the first step.
3	• The Applicant identifies himself/herself by presenting a temporary secret that was previously set or sent to the physical address on record. • Permanent secrets are only issued within protected sessions. For physical transactions, • The Applicant identifies himself/herself in person using a secret as described above, or through the use of a biometric recorded previously. • Temporary secrets are not reused.
4	Only physical transactions apply. The Applicant shall identify himself/herself in person for physical transactions previously captured by biometrics.

R and IdP process occurs before any authentication instance between two entities.

3. Authentication Tokens

A token is defined as something the user owns and controls (typically a cryptographic module or password) that is used to authenticate his/her identity. A token typically contains a secret value, or token secret, used to generate authenticator outputs on demand to verify the user's possession of the token. Tokens are based on one or more authentication factors [16]:

- Something you know (knowledge tokens): such as a password.
- Something you have (ownership tokens): such as an ID card.
- Something you are (identity tokens): such as a person's fingerprint.

Hence, a token can be hardware, software, or some information you remember. Tokens are categorized into two classes based on the number of factors. They are

- Single-factor token, which depends on one authentication factor, such as a password.
- Multi-factor token, which employs two or more authentication factors, such as a smart card and a Personal Identification Number (PIN) to activate it.

There are many types of tokens:

- Memorized Secret Token
- Pre-registered Knowledge Token.
- Look-up Secret.
- Out of Band.
- Single-factor (SF) One-Time Password (OTP) Device.
- Single-factor (SF) Cryptographic Device is an ownership token.
- Multi-factor (MF) Software Cryptographic.
- Multi-factor (MF) OTP.
- Multi-factor (MF) Cryptographic Device.

Table A2 lists LoA values for various single-factor token types [16]. In addition, there are specifications for the Verifier and the token for each type of token.

Table A2. LoA components for Token Types [16]

Token Type	LoA component	Token Requirements	Verifier Requirements
Memorized Secret Token	Level 1	PWD length >= 6 chars chosen from an alphabet of >= 90 chars or a randomly generated PIN of 4 or more digits	-Limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
	Level 2	PWD length >= 6 chars chosen from an alphabet of >= 90 chars, or a randomly generated PIN of 6 or more digits.	- Implement composition rules to constrain user options. - Limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
Pre-Registered Knowledge Token	Level 1	The secret provides at least 14 bits of entropy.	-Limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
		The entropy cannot be directly calculated, as in the case of a user chosen password or personal knowledge questions.	-If the questions are not supplied by the user, the user selects prompts from a set of at least five questions.

			-Empty answers are prohibited. -Answers are verified for at least three questions - Limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
	Level 2	The secret provides at least 20 bits of entropy.	-Limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
Pre-Registered Knowledge Token	Level 2	The entropy in the secret cannot be directly calculated, as in the case of a user chosen password or personal knowledge questions.	-If the questions are not supplied by the user, the user selects prompts from a set of at least seven questions. -Empty answers are prohibited. - Answers are verified for at least three questions. - Limit the number of failed authentication

			attempts to 100 or less in any 30-day period.
Look-up Secret Token	Level 2	The token authenticator has 64 bits of entropy.	N/A
		The token authenticator has at least 20 bits of entropy.	limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
Out of Band Token	Level 2	The token is uniquely addressable and supports communication over a channel apart from the primary channel for E-Authentication.	-The Verifier generated secret has at least 64 bits of entropy.
			-The Verifier generated secret has at least 20 bits of entropy. - Limit the number of failed authentication attempts to 100 or less in any 30-day period of time.
SF One-Time Password Device	Level 2	Use an Approved block cipher or hash function to combine a symmetric key stored on the device with a nonce to generate an OTP.	- The OTP has a limited lifetime (minutes). - The cryptographic module performing the Verifier function is validated at US Federal Information Processing

		The nonce may be a date and time, or a counter generated on the device.	Standards (FIPS) 140-2 Level 1 or higher.
SF Cryptographic Device	Level 2	The cryptographic module is validated at FIPS 140-2 Level 1 or higher.	Verifier generated token input has at least 64 bits of entropy.
MF Software Cryptographic Token	Level 3	- The cryptographic module is validated at FIPS 140-2 Level 1 or higher. - Each authentication requires entry of the password or other activation data. - The unencrypted copy of the authentication key is erased after each authentication.	Verifier generated token input has at least 64 bits of entropy.
MF OTP Hardware Token	Level 4	- Cryptographic module is FIPS 140-2 validated at Level 2 or higher, with physical security at FIPS 140-2 Level 3 or higher. - The OTP is generated using an Approved block cipher or hash function to combine a symmetric key stored on a personal hardware device with a nonce - The nonce may be a date and time, or a counter generated on the device.	The OTP has a lifetime < 2 minutes.

		- Each authentication requires entry of a password or other activation data through an integrated input mechanism.	
MF Hardware Cryptographic Token	Level 4	- Cryptographic module is FIPS 140-2 validated at Level 2 or higher; with physical security at FIPS 140-2 Level 3 or higher. - Requires the entry of a PWD, PIN, or biometric to activate the authentication key. - Export of authentication keys is prohibited.	Verifier generated token input (e.g., a nonce or challenge) has at least 64 bits of entropy.

Entropy denotes the uncertainty in the value of a secret (such as a password) and is conventionally expressed in bits. It is used to measure the difficulty in guessing the secret.

Table A2 summarizes the requirements for single-token authentication. As for multi-token authentication , the following rules are applied [16]:

- To achieve level 3 assurance, two tokens of level 2 assurance need to be employed, one of which is of the ownership type and the other of the knowledge type.
- Combining two types of tokens produces a multi-factor token at the highest level of the two combined.

4. Token and Credential Management

A credential is an object that authoritatively binds an identity to a token possessed and controlled by a user. Credentials and tokens need to be managed to reflect any changes in their values at binding to maintain the same level of assurance. Token and credential management activities are performed by the Credential Provider CP, and they include the following [16]:

- Credential storage: The CP may store credentials. Private credentials require extra confidentiality mechanisms as opposed to public credentials. Weakly bound credentials also require extra integrity checks, unlike strongly bound credentials.
- Token and credential verification services: When the Verifier and the CP are not part of the same entity, the CP is responsible for providing the Verifier with the information needed to facilitate the credential verification process, such as checking a password against a PWD database, for example.
- Token and Credential renewal and re-issuance: Renewal means the extension of the validity period of a token or a credential without changing the identity. Re-issuance, on the other hand, requires creating new credentials for a new identity. The CP enforces suitable policies for renewal and re-issuance, such as setting expiration periods and grace periods for expired tokens.
- Token and credential revocation and destruction: The CP is responsible for maintaining the revocation status of credentials and destroying the credential at the end of its life. "Public credentials" usually require an explicit revocation mechanism, which may have with a preset validity period, while "private credentials" revocation and destruction are implemented through an update of CP's local credential storage [16].
- Records retention: The CP maintains a record of the registration, history, and status of tokens and credentials, including revocation. For level 2 and higher, a minimum record retention period is required.
- Security controls: The CP also implements and maintains the appropriate security controls as per NIST SP 800-53 publication [16].

The AoL component values for some or all of the token and credential management activities are stipulated by NIST [16] as indicated in the below Table A3.

Table A3. LoA Requirements for Token and Credential Management Activities

	Requirements			
Token and Credential Management Activities	Level 1	Level 2	Level 3	Level 4
Storage	-Limit files access to administrator; -Passwords are stored inverted or as a one-way hash	- Limit files access to administrators -Passwords are stored encrypted or hashed (after being concatenated to a variable salt)	-Limit files access to administrators. -FIPS 140-2 level 2 encryption or higher for files and keys	
Verification Services	Long term token secrets are not to be shared with other parties unless necessary	- Long-term shared secrets are only shared with Verifiers operated by CP -Cryptographic protection for all messages between CP and Verifier for private credentials -Private credentials are sent only through a protected session	-CPs provide a secure mechanism to allow Verifiers or CSPs to ensure credentials validity. -Temporary session authentication keys may be generated from long-term shared secret keys by CPs and distributed to third party Verifiers.	

Renewal and Reissuance	N/A	-Proof-of-possession of the unexpired current token needed is required for renewal and re-issuance. -Passwords cannot be renewed; only re-issued. -Upon re-issuance, the token secrets are not to be set to a default or reused value in any manner. -All interactions occur over a protected session	- Only allowed prior to the expiration of the current credential. - All interactions occur over a protected session	-Sensitive data transfers to be cryptographically authenticated using keys bound to the authentication process. -All short-term keys derived during the original authentication operation expire and must be re-authenticated after no more than 24 hours from the initial authentication.
Revocation and Destruction	N/A	CPs revoke credentials and tokens within 72 hours after being notified that a credential is no longer valid	CPs revoke credentials and tokens within 24 hours after being notified that a credential is no longer valid	- CPs revoke credentials and tokens within 24 hours after being notified that a credential is no longer valid - Verifiers or CSPs ensure that the credentials are

				freshly issued (within 24 hours)
Records Retention	N/A	-For seven years and six months beyond the expiration or revocation of a Credential.		-For ten years and six months beyond the expiration of Credential.
Security Controls	N/A	CP must employ appropriately tailored security controls from the low baseline of security controls defined by NIST		

5. Authentication Process

The goal of the authentication process is for the Verifier to establish the identity of a claimant through an authentication protocol message exchange, during or after which a protected session is established for further data exchange. There are a number of threats against the authentication process, which require management mechanisms at both ends of the Verifier and the user. These threats include Online guessing, Phishing, Pharming, Eavesdropping, Replay, Session hijacking, Man-in-the-middle, and Denial of Service DOS attacks. Table A4 indicates the LoA values for level of resistance against known threats:

Table A4. Requirements for Authentication Process Threat Resistance for LoA [16]

Authentication Process Threats	Threat Resistance Requirements			
	Level 1	Level 2	Level 3	Level 4
Online guessing	Yes	Yes	Yes	Yes
Replay	Yes	Yes	Yes	Yes
Session hijacking	No	Yes	Yes	Yes
Eavesdropping	No	Yes	Yes	Yes
Phishing/Pharming (Verifier impersonation)	No	No	Yes	Yes
Man in the middle	No	Weak	Weak	Strong
Denial of service/flooding	No	No	No	No

To satisfy the threat resistance as per Table A4, suitable authentication technologies must be used. Table A5 lists examples of technologies sufficient for meeting each LoA requirements [16].

Table A5. Examples of Technologies Sufficient for LoA

Examples of Authentication Technologies	LoA component value
- Password with a challenge-response protocol. - Password sent over a TLS - Password-based versions of Kerberos. - APOP	Level 1

- S/KEY	
- Password through a secure encrypted TLS protocol session.	Level 2
- Client-authenticated TLS (implemented in all modern browsers), for claimants with public key certificates. - Tunneling the output of an MF OTP Token, or the output of an SF OTP Token in combination with a Level 2 personal password, through a TLS session.	Level 3
- Client-authenticated TLS (implemented in all modern browsers), with claimants who have public key MF Hardware Cryptographic Tokens. - The token used for strong man-in-the-middle resistance need not be hardware token.	Level 4

6. Assertions

Assertions are statements from a Verifier to a CSP containing identity information and verified attributes about a user who was successfully authenticated [16]. There are three models of authentication with varying types of assertions as follows [16]:

- **The Direct Model**: In this model, the user presents his token to authenticate to the Verifier to receive an assertion that is forwarded to the CSP.
- **The Indirect Model:** In this model, the claimant uses his token to authenticate with the Verifier. After successful authentication, the Verifier creates an assertion and an assertion reference (which identifies the Verifier and includes a pointer to the full assertion). The assertion reference is then sent to the subscriber to be forwarded to the CSP for authentication. The CSP uses the assertion reference to request the assertion explicitly from the Verifier.

Based on the content of both direct and indirect models, assertions are categorized into two types [16]:

- **Holder-of-Key Assertion**: This type contains a reference to a symmetric key or a public key possessed by the subscriber. For this type, the CSP may require the subscriber to prove possession

of the referenced secret, which to a degree proves his rightful ownership of the assertion. Thus, this type is secure since an attacker can't easily prove possession of the secret referenced in the assertion.

- **Bearer Assertion**: This type of assertion does not provide a way for the claimant to prove rightful ownership of the assertion. In the direct model, if a bearer assertion or assertion reference is captured, copied, or manufactured by an attacker, it may lead to impersonation. Hence, this type of assertion is secure only if a part of the assertion or assertion reference can be kept secret and unpredictable by attackers.

- **The Proxy Model:** In this model, the user uses his token to authenticate with the Verifier. After successful authentication, the Verifier creates an assertion and includes it when interacting directly with the CSP. The CSP then grants or denies the request based on the assertion made by the Verifier. Verifiers may also pass along useful information to the CSP, such as device identity, location, system health checks, and configuration management. Communications between the Verifier and the CSP should be protected. There are three types of assertion technologies used for this model [16]:

- **Cookies**: They are the text files used by browsers to store information provided by a particular website and are sent back to the site each time the browser requests a page to identify the user, provide customization, or authorize the user for transactions. Cookies have two compulsory parameters (name of the cookie and the value stored in the cookie) and four optional parameters (the expiration date, the path, the domain, and a flag for a secure connection).

- Security Assertion Markup Language (SAML) Assertions: It uses an XML-based framework for creating and exchanging authentication and attribute information between trusted entities over the Internet. The components of SAML include the Assertions XML schema for defining assertion structure, the SAML Protocols used to request assertions and assertion references, and the bindings that define the underlying communication protocols (such as HTTP or SOAP) used to transport the SAML assertions. SAML Assertions may contain three types of statements (authentication statements, attribute statements related to the subscriber, and authorization statements for resources the subscriber is permitted to access).

- **Kerberos Tickets**: They are used to support authentication of a claimant over a shared network using two or more Verifiers. The claimant implicitly authenticates to the Verifier by decrypting a random session key encrypted during registration. The Verifier additionally generates

a Kerberos ticket. The Kerberos ticket is an encrypted object containing the same session key, the identity of the owner, and an expiration time. The ticket's confidentiality and integrity are also ensured.

NIST guidelines list a number of threats against assertions [16]. Table A6 lists these threats and states the LoA component values for threat resistance to be provided against each threat.

Table A6. Threats against Assertions per LoA [16]

Threat	Description	Level 1	Level 2	Level 3	Level 4
Assertion manufacture/ modification	An attacker generates a fake assertion or modifies the assertion content to his advantage.	Yes	Yes	Yes	Yes
Assertion disclosure	Disclosure of information within the assertion, making the subscriber open to other attacks.	No	Yes	Yes	Yes
Assertion repudiation by Verifier	This occurs when the Verifier repudiates the assertion due to an improper mechanism such as an unsigned assertion.	No	No	Yes	Yes
Assertion repudiation by subscriber	When the subscriber repudiates any transaction with the CSP that was authenticated using a bearer assertion.	No	No	No	Yes
Assertion redirect	An attacker uses the assertion intended for one CSP to obtain access to another.	No	Yes	Yes	Yes

Assertion reuse	An attacker attempts to use an assertion that has already been used once.	Yes	Yes	Yes	Yes
Secondary authenticator manufacture	An attacker attempts to generate a valid secondary authenticator and uses it to impersonate a subscriber.	Yes	Yes	Yes	Yes
Secondary authenticator capture	An attacker uses a session hijacking attack or a man-in-the-middle attack to capture the secondary authenticator when the Verifier transmits it to the subscriber or as it is being used by the subscriber to authenticate to the CSP. The secondary authenticator can be used to impersonate the subscriber.	No	Yes	Yes	Yes
Assertion substitution	A subscriber attempts to impersonate a more privileged subscriber by subverting the communication channel between the Verifier and CSP.	No	Yes	Yes	Yes

Thus, for each assurance Level, there are minimum requirements to provide threat resistance as per Table A6. Table A7 lists the requirements for each LoA to provide protection against the defined set of threats [16].

Table A7. Requirements for Assertions at each LoA

Assurance Level	Requirements
Level 1	-It must be impractical for an attacker to manufacture an assertion or assertion reference. -For the direct model, the assertion is signed by the Verifier using a secret key shared by the Verifier and CSP. -For the indirect model, the assertion reference has a minimum of 64 bits of entropy. -Bearer assertions and assertion references are generated for one-time use. - All assertions from the Verifier to the CSP should either be signed by the Verifier or transmitted via a protected session. -The CSP must establish a binding relationship between the assertion reference and its corresponding assertion via signed communications with the authenticated Verifier. -Single-domain Assertions expire within 5 minutes. -Cross-domain Assertions may last up to 12 hours.
Level 2	-Assertions and Assertion references are protected against manufacture/modification, capture, redirect, and reuse. -Each assertion is for a single CSP. -Assertions, assertion references, and any session cookies are transmitted via a protected session and linked to the primary authentication process to avoid session hijacking attacks. -Assertions sent from the Verifier to the CSP are sent via a mutually authenticated protected session or signed by the Verifier and encrypted.

	-All assertion protocols used at Level 2 and above require the use of Approved cryptographic techniques. -The use of Kerberos keys derived from user-generated passwords is not allowed.
Level 3	-Assertions are protected against repudiation using signatures and verified names rather than pseudonyms. -Kerberos tickets are acceptable as assertions level 3 under the following conditions: •All Verifiers operate under a single management; • The subscriber authenticates to the Verifier using a Level 3 token. • All level 3 requirements unrelated to non-repudiation are satisfied. -Single-domain assertions expire within 30 minutes. -Cross-domain assertions expire if not used within 5 minutes. -To achieve single sign-on, Verifiers can re-authenticate the subscriber prior to delivering assertions to new CSPs, using a combination of long term and short-term single domain asertions given that: • The subscriber has successfully authenticated within the last 12 hours. • The subscriber can demonstrate that he or she was the party authenticated with the Verifier. • The Verifier proves that the subscriber was not idle for more than 30 minutes.
Level 4	-Bearer assertions and cookies are not used to establish the identity but can be used to bind keys or other attributes to it -Holder-of-key assertions may be used provided that:

	• The claimant authenticates with the Verifier using a Level 4 token in a Level 4 authentication protocol. • The key referenced in the assertion is Level 4 token; • The CSP verifies that the subscriber possesses the key referenced in the assertion using a level 4 protocol. -The CSP should maintain records of the assertions it receives to compare values with the CP in case of suspicious transactions at the CSP. - Kerberos tickets are acceptable as assertions at level 4 provided that: • All are under a single management authority; • The subscriber authenticates to the Verifier using a level 4 token; • All Level 4 requirements unrelated to non-repudiation are satisfied.

Based on the detailed NIST LoA component derivations stated above, Table A7. shows examples of some component values for Run-Time LoA-related factors.

Appendix B

Implementation for algorithm performance evaluation

Implementation of sorting LoA determine minimum LoA value (LowSort.java)

```
import java.io.*;
import java.util.Scanner;

public class LowSort {
   //Array of LoA values of dependent factors
   static int [] loaArray = {5,5,5,5,5} ;
   static int i = 0;


      public static void main (String [] args){


         // Read LoA values of dependent factors
         try {
          Scanner console = new Scanner(new File("in.txt"));


          while (console.hasNextLine()) {
             loaArray[i] = console.nextInt();
             i++ ;
             }
          }catch (IOException e) {
             System.out.println("error reading file! ");
          }


         // Sort LoA array
         int min = 0;
         for(int i=0;i<4;i++){
```

```
            if (loaArray[i+1] < loaArray[i] ){
                min = loaArray[i+1];
                loaArray[i+1] = loaArray[i];
                loaArray[i] = min;}
          min = loaArray[0];
    }
}
}
```

Implementation of pairwise comparisons of LoA values determine minimum (LowCompare.java)

```
import java.io.*;
import java.util.Scanner;

public class LowCompare {
   //Array of LoA values of dependent factors
   static int [] loaArray = {5,5,5,5,5} ;
   static int i = 0;

      public static void main (String [] args){
         // Read LoA values of dependent factors
         try {
          Scanner console = new Scanner(new File("in.txt"));

          while (console.hasNextLine()) {

             loaArray[i] = console.nextInt();
             i++ ;
             }
```

```
        }catch (IOException e) {
            System.out.println("error reading file! ");
        }

        // compare
        int min = loaArray[0];
        for(int i=1;i<=4;i++){
            if (loaArray[i] < min )
                min = loaArray[i];}
    }
}
```

Implementation of Additive Rule based on Probability theory algorithm (LoaPadd.java)

```
import java.io.*;
import java.util.Scanner;
public class LoaPadd {

   //Array of LoA values of independent factors
   static int [] loaArray ;
   static int AloA;
   static int i = 0;
   static int temp = 0;

      public static void main (String [] args){

        // Read LoA values of independent factors
        try {
         Scanner console = new Scanner(new File("in.txt"));
```

```
          while (console.hasNextLine()) {
             loaArray[i] = console.nextInt();
             i++ ;
             }
          }catch (IOException e) {
             System.out.println("error reading file! ");
          }

         // determine ALoA based probability theory
         if (i == 1) {
            AloA =  loaArray[0];}    // In case of one factor
         else  {        // Otherwise, apply inclusion-exclusion principle
            temp = 1;
            for (i=0;i<4;i++) {
               temp = temp * (1- (loaArray[i]/4));} // P (ri) = LoA /4
            temp = temp -1;
            AloA = temp;}
      }
}
```

Implementation of Additive Rule based on Mapped Weights algorithm (LoaMadd.java)

```
import java.io.*;
import java.util.Scanner;
public class LoaMadd {

   //Array of LoA values of independent factors
   static int [] loaArray = {0,0,0,0,0} ;
   static int AloA;
   static int i = 0;
```

```
static int temp = 0;
static int w;

   public static void main (String [] args){
      // Read LoA values of independent factors
     try {
      Scanner console = new Scanner(new File("in.txt"));


      while (console.hasNextLine()) {
         loaArray[i] = console.nextInt();
         i++ ;
         }

      }catch (IOException e) {
         System.out.println("error reading file! ");
      }

     // determine ALoA based on mapped weight algorithm
     if (i == 1) {
       AloA =  loaArray[0];}    // In case of one factor
     else  {        // Otherwise, apply inclusion-exclusion principle
       for (i=0;i<4;i++) {
         temp = temp + (1- (loaArray[i]/4));} // // Adjust loa values with weights

       AloA = temp;}
   }
}
```

Implementation of Additive Rule based on Combined LoA algorithm (LoaCadd.java)

```
import java.io.*;
import java.util.Scanner;
public class LoaCadd {

   static int [] loaArray = {0,0,0,0,0} ; // LoA values of independent factors
   static int [] loaCount = {0,0,0,0} ; // count of equivalent LoA component values
   static int AloA;
   static int i = 0;

      public static void main (String [] args){

         // Read LoA values of independent factors
         try {
          Scanner console = new Scanner(new File("in.txt"));

          while (console.hasNextLine()) {
             loaArray[i] = console.nextInt();
             i++ ;
             }

          }catch (IOException e) {
             System.out.println("error reading file! ");
          }


         // determine ALoA based on combined LoA algorithm
          if (i == 1) {
            AloA =  loaArray[0];}    // In case of one factor
          else  {
            AloA = 1;
            for (i=1;i<4;i++)    // for LoA component values to be possibly combined (1 to 3)
               for (int x =1;x<4;x++){
```

```
                if (loaArray[x] == i) {
                    loaCount[i]++ ; }

              if (loaCount[i] >= 2) {   // when 2 or more factors have LoA = i
                 loaCount[i+1]++;
                 AloA = i+1;}
               }
             }
       }
}
```