A Structured Approach to Electronic Authentication Assurance Level Derivation

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Science

2010

Li Yao
School of Computer Science

CONTENTS

Final words count: 33,540

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

Agg-LoA. ..... Aggregate LoA
AHP ..... Analytic Hierarchy Process
ALoA-DM ..... Authentication LoA Derivation Module
GEA-LoADM ..... Generic E-authentication LoA Derivation Model
GSI ..... Grid Security Infrastructure
LoA ..... Authentication Levels of Assurance
LoA-ACM ..... LoA-effecting Attributes Collection Module
LoA-AHS ..... LoA-effecting attribute hierarchical structure
LoA-APDB ..... LoA-effecting attributes policy database
LoA-APM ..... LoA-effecting Attributes Policy Manager
LoA-AWAM ..... LoA-effecting Attributes Weightings Allocation Module
MAIS ..... Multi-Agency Information Sharing System
MCDM ..... Multi-Criteria Decision Making
NIST ..... US National Institute of Standards and Technology
OCR ..... online credential repository
PC ..... Proxy Certificate
PKI ..... Public Key Infrastructure
RA ..... Registration Authority
SAML ..... Security Assertions Markup Language
SP ..... Service Provider
SSO ..... Single Sign-On

ABSTRACT

We envisage a fine-grained access control solution that allows a user's access privilege to be linked to the confidence level (hereafter referred to as the assurance level) in identifying the user. Such a solution would be particularly attractive to a large-scale distributed resource sharing environment, where resources are likely to be more diversified and may have varying levels of sensitivity and resource providers may wish to adjust security protection levels to adapt to resource sensitivity levels or risk levels in the underlying environment. However, existing electronic authentication systems largely identify users through the verification of their electronic identity (ID) credentials. They take into account neither assurance levels of the credentials, nor any other factors that may affect the assurance level of an authentication process, and this binary approach to access control may not provide cost-effective protection to resources with varying sensitivity levels.

To realise the vision of assurance level linked access control, there is a need for an authentication framework that is able to capture the confidence level in identifying a user, expressed as an authentication Level of Assurance (LoA), and link this LoA value to authorisation decision-making. This research investigates the feasibility of estimating a user's LoA at run-time by designing, prototyping and evaluating an authentication model that derives an LoA value based upon not only users' ID credentials, but also other factors such as access location, system environment and authentication protocol used.

To this aim, the thesis has identified and analysed authentication attributes, processes and procedures that may influence the assurance level of an authentication environment. It has examined various use-case scenarios of authentication in Grid environments (a well-known distributed system) and investigated the relationships among the attributes in these scenarios. It has then proposed an authentication model, namely a generic e-authentication LoA derivation model (GEA-LoADM). The GEA-LoADM takes into account multiple authentication attributes along with their relationships, abstracts the composite effect by the multiple attributes into a generic value called the authentication LoA, and provides algorithms for the run-time derivation of LoA values. The algorithms are tailored to reflect the relationships among the attributes involved in an authentication instance. The model has a number of valuable properties, including flexibility and extensibility; it can be applied to different application contexts and supports easy addition of new attributes and removal of obsolete ones.

The prototypes of the algorithms and the model have been developed. The performance and security properties of the LoA derivation algorithms and the model are analysed here and evaluated based on the prototypes. The performance costs of the GEA-LoADM are also investigated and compared against conventional authentication mechanisms, and the security of the model is tested against various attack scenarios. A case study has also been conducted using a live system, the Multi-Agency Information Sharing (MAIS) system.

DECLARATION

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

COPYRIGHT

ACKNOWLEDGEMENTS

First of all, I would like to express my sincerest gratitude to my supervisor, Dr. Ning Zhang, for her guidance, valuable advice and encouragement throughout this research. This research would not have been possible without her whole-heartedly support.

I would also like to thank my wife Qing Li, for her love, sacrifice and patience during the period. Without her support this research would have been much difficult to complete.

My thanks go to my colleagues from the security group. Their warmth and friendship have provided a pleasant environment during my years of study at The University of Manchester.

DEDICATION

To *mum* and *dad*; I love you.

Chapter 1

Introduction

1.1 Electronic Authentication (E-Authentication)

Electronic authentication (e-authentication) is an electronic process by which a claimant (i.e. an individual or a software component) can be identified by a verifier (i.e. a party that establishes the identity of the claimant). It provides assurance as to whether someone or something is who or what they claim to be in a digital environment. Thus, e-authentication plays a key role in the establishment of trust relationships for electronic commerce, electronic government and many other social interactions. It is also an essential component of any strategy for protecting resources (services and data), information systems and networks and other assets from unauthorised access or identity theft. E-authentication is therefore vital for establishing accountability online and for securing resource sharing in distributed environments.

A typical e-authentication procedure consists of three key phases. The first phase is user registration/user identity proofing and credential issuance. The second phase is a claimant authentication process and the third phase is credential renewal/revocation/destruction. For example, if a user wishes to access certain services from a service provider (SP) he/she first needs to register with a Registration Authority (RA), which could be the SP itself or a third party that provides the credential management service for the SP. Depending on the type of service that the SP provides, the RA may also need to demonstrate that the identity is a real identity and that the user is the person who is entitled to use that identity (i.e. user identity proofing). For example, if a user applies for a free e-mail service, identity proofing is usually unnecessary; whereas a strict identity proofing process is required if a user applies for an online banking service. After successful user registration, the user can subsequently be issued a credential. The credential could either be something he/she knows (e.g. a pin or a password), or something he/she has (e.g. a smart card or a USB token with a PKI credential stored in it). The user (claimant) proves his/her identity to the verifier by demonstrating that he/she

processes a valid credential. When the credential reaches its expiration time, it should be renewed if the user requires further access to the SP, or destroyed if not. The credential can be revoked if it is, or is suspected to be, compromised.

1.2 Authentication Levels of Assurance (LoA)

Different types of credentials may provide different authentication levels of assurance. LoA is defined as the *strength of authentication* required for a service provider to be assured that resource access is only granted to users whose identities have been verified. It reflects the degree of confidence in an authentication process used to establish the identity of a claimant to whom the credential was issued and the degree of confidence that the claimant using the credential is indeed the entity to whom the credential was issued.

In fact, all the factors in the three phases of the e-authentication procedure influence the LoA established. In addition to the type of authentication credential (or authenticator or token) used by the claimant, such factors also include the authentication protocol used by the underlying authentication service and how the credentials are managed. The latter encompasses issues such as the token technology that is used to store the credential, the manner in which a claimed identity is bound to an authentication credential, the life-cycle management of the credential and whether the credential service provider (CSP) has sufficient operating procedures, processes and policy frameworks to establish the required level of trust. Furthermore, the extent to which an authentication event is coupled to an authorisation event should also be taken into account when the LoA is established. These authentication factors, processes and procedures mentioned above can collectively be called *LoA-effecting attributes*. In other words, we now have the following definition.

***Definition 1 - LoA-effecting attributes*:** An LoA-effecting attribute refers to a factor that plays a part in the value of an authentication level of assurance associated with an authentication instance (an authentication instance is an authentication event that takes place in an authentication environment). For example, authentication credential/token types, authentication protocol/mechanisms and the processes/procedures in managing

users' credentials are all examples of LoA-effecting attributes.

Once the LoA-effecting attributes are identified, their individual contributions towards the overall authentication level associated with an authentication instance should be quantified or determined. To this end, we have two further definitions.

***Definition 2 – A component LoA (denoted as LoA_x)*:** A component LoA, LoA_x, refers to the LoA value contributed by LoA-effecting attribute x. For example, $LoA_{AuthNMethod}$ is used to denote the LoA contribution made by the authentication method used in an authentication instance.

***Definition 3 – An aggregate LoA (denoted as Agg-LoA)*:** The Agg-LoA refers to the overall LoA value determined by the combined effects of multiple LoA-effecting attributes involved in an authentication instance. For example, if there are n LoA-effecting attributes $\{a_1, a_2, ..., a_n\}, n > 0$ in an authentication instance, each with a component LoA, LoA_i where $i \subseteq [1, n], n > 0$, then the aggregate LoA for this authentication instance is determined by the component LoA values of the n attributes $\{LoA_1, LoA_2, ...LoA_n\}$. In other words, given component LoA values, the aggregate LoA associated with an authentication instance can be determined by the following equation:

$$\text{Agg-LoA} = f(\text{LoA}_1, \text{LoA}_2, \ldots, \text{LoA}_n) \qquad (1.1)$$

where *f* is a function capturing the relationship among the n attributes.

1.3 Research Motivation

In a typical access control process, once a user has made an access request, the user is authenticated then an authorisation process is invoked to make a grant or deny decision on the access request. The authorisation decision is usually based on the user's identity, his/her membership or role and access permission, which is typically expressed as 'what action on which object'. In other words, an access control policy is usually expressed using the tuple <subject/identity, role/membership, action, object>. If a user is authenticated and the access request made by the user conforms to the specified access control policy, the authorisation process will make a grant decision and the access will be

permitted. Otherwise, it will be denied. This approach to access control is a binary approach. A grant or deny authorisation decision is made based merely upon the outcome of a credential-based authentication process. It does not quantify the assurance level of the process. It is well known that identity authentication cannot always produce a perfect and reliable outcome [Gang01]. Different authentication instances in different authentication environments, as influenced by the use of different credentials, authentication protocols and/or token management procedures, may result in different levels of assurance. These different LoAs should be taken into account when an access request is being granted. This is because in a large-scale distributed resource sharing environment, resources are likely to be more diversified and may have varying levels of sensitivity, or resource providers may wish to adjust security protection levels to adapt to resource sensitivity levels or the risk levels in the underlying environment. The existing approach to access control, disregarding the quality of authentication in authorisation decision-making, cannot satisfy the need for an effective and cost-efficient security provision in the diversified resource sharing environments. For example, if we opt for a simpler or cheaper authentication method that satisfies the authentication requirements of less-sensitive resources, then the more-sensitive resources will be put at a higher risk. On the other hand, if we choose a more stringent authentication method to satisfy the authentication requirements related to more-sensitive resources, access to the less-sensitive resources will entail unnecessary higher costs in performance and usability as more secure credentials and a more stringent authentication procedure usually introduces a higher level of performance and usability costs. To address this weakness and to optimise performance and usability while at the same time providing an adequate level of protection, there is a need for an authentication framework that can capture the confidence level in identifying a user, expressed as an authentication LoA, and link this LoA value to authorisation decision-making.

1.4 Research Challenges

There are a number of challenges that must be overcome when designing such a framework. Firstly, there are typically multiple attributes influencing the value of an aggregate LoA in an authentication instance. In addition, the number and types of LoA-

effecting attributes may differ in different authentication contexts/environments. The framework designed should be able to accommodate this diversified feature. Also, as technology advances, there may be new emerging LoA-effecting attributes that should to be taken into account or obsolete ones taken out of the equation when evaluating an aggregate LoA value, so the framework should allow for an easy addition/deletion of new/obsolete LoA-effecting attributes. Secondly, the impacts (i.e. weightings) of different LoA-effecting attributes on the aggregate LoA may vary and those of the same LoA-effecting attribute may change over time or in different authentication contexts/environments. In real life, it should be the access control policy makers who decide the relative weightings of multiple LoA-effecting attributes in their authentication process. Such decisions are dependent on a number of factors, including their access control policy, how authentication services are provided, the trustworthiness of the authentication service, etc. Basically, the more reliable and trust-worthy an LoA-effecting attribute, the higher the impact or greater the contribution it should have towards the aggregate LoA. Thirdly, the relationship between different attributes may vary. Some attributes may jointly reinforce the overall authentication assurance level (i.e. in an additive relationship), while others may follow the weakest link principle. Figure 1.1 outlines the challenging issues for designing the framework as discussed above and illustrates the methods used to address these challenging issues.

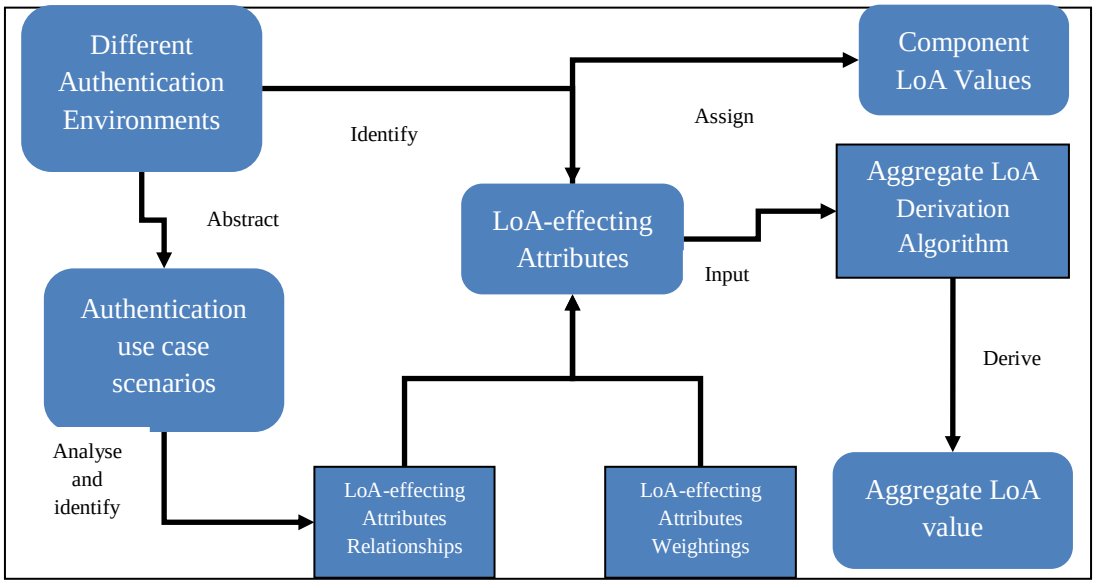

Figure 1.1 Research methodology and design challenges

1.5 Research Aim and Objectives

We envisage creating a fine-grained access control solution that allows a user's access privilege to be linked to the assurance level in identifying the user. For example, if a user requests access for a resource object with a higher sensitivity level, and/or the risk level in the underlying access environment is higher, we could impose a higher LoA value which the user has to satisfy before he can be granted access. The use of LoA in access control decision making acknowledges the fact that the more LoA-effecting attributes that are considered or used in identifying a user, the higher the reliability and the confidence level we will have in the authentication outcome, thus the more privileges the system could grant to the user. Furthermore, the more the resource is worth and/or the higher the risks in the underlying access environment, the more stringent the authentication process should be. Such a solution can provide more fine-grained access control than the existing binary approach mentioned earlier. It would be particularly attractive to large-scale distributed resource sharing environments where resources under protection are more likely to be diversified.

The aim of this research is to set the first step to achieving this vision of LoA-linked access control: that is to design, prototype, and evaluate an authentication framework that takes into account not only users' ID credentials, but also other LoA-effecting attributes, such as authentication mechanism, access environment, authentication protocol used, and derives the LoA value of a user in a given authentication instance based upon the set of LoA-effecting attributes involved in the instance.

To fulfil this aim, the following objectives have been defined:

1. To investigate and critically analyse related work in network/distributed system authentication and authentication level of assurance.
2. To identify and study different authentication scenarios and models used in distributed systems with the aim of identifying factors or parameters (i.e. LoA-effecting attributes) that may affect the authentication LoA.
3. To study the relationships among multiple LoA-effecting attributes in various authentication use-case scenarios and models and to devise algorithms to capture the

aggregated effects in these scenarios and models.

4. To design an authentication framework that, for a given set of LoA-effecting attributes in an authentication instance, could derive an aggregate LoA value in real-time based upon individual LoA values of the attributes, their respective impacts and their mutual relationships.
5. To prototype the model designed.
6. To conduct testing, analyses and case studies to evaluate the security and performance of the model using the prototype.

1.6 Achievements and Novel Contributions

This research has made the following major advances. The first is the design, prototyping and evaluation of aggregate LoA derivation algorithms. The second is the design, implementation and evaluation of a generic e-authentication LoA derivation model (GEA-LoADM). More details of these novel contributions are given below.

Aggregate LoA Derivation Algorithms

The first novel contribution is the identification of two relationships among LoA-effecting attributes and the design of algorithms that capture these two relationships and derive an aggregated authentication LoA value given a set of LoA-effecting attributes in each of the relationships.

One of the two relationships is the weakest link relationship (WLR), and the other is the additive relationship (AR). The WLR algorithm captures the relationship among a set of LoA-effecting attributes that are included in an authentication instance in the form of *n* operational chain. The aggregate LoA of the attribute set in the WLR is equal to the lowest component LoA of individual attributes. The AR algorithm captures the relationship among a set of LoA-effecting attributes that are involved in an authentication, where their component LoA values reinforce each other. The aggregate LoA of this attribute set should not be less than the highest value of the component LoA value of the attribute set.

LoA-effecting attribute hierarchical structure

The second major novel contribution is the design of the LoA-effecting attribute hierarchical structure (AHS). As discussed in Section 1.4, the number and types of LoA-effecting attributes involved in an authentication process are application dependent. Different authentication models or use-case scenarios may involve the use of a different set of LoA-effecting attributes. In addition, the same user being authenticated in two different authentication instances may also lead to two different sets of such attributes. This is because they are also dependent on access policies that are, in turn, influenced by factors such as asset values and the underlying risks in the access environment. In this regard, the author has investigated and extended the attributes identified by NIST [Burr06] and OASIS [Samlac] and, produced a set of generic LoA-effecting attributes. Furthermore, the author has examined the mutual relationships among these attributes and created a hierarchical structure that can accommodate these attributes in a systematic manner. The significance of this AHS is that it provides a structure for systematic and run-time determination of an aggregate LoA value given a set of LoA-effecting attributes.

Generic e-authentication LoA derivation model

The third major novel contribution is the design of the GEA-LoADM model that incorporates aggregate LoA algorithms and the AHS structure. The model is capable of collecting LoA-effecting attributes, retrieving the component LoA value of each attribute and deriving an aggregate LoA value for an authentication instance in real time.

Publications:

- Li Yao and Ning Zhang, "Quantifying Authentication Levels of Assurance in Grid Environments", the 6th Information Assurance and Security Conference 2010 (IAS2010), Aug 23-25, 2010, Atlanta, USA, pp 298-303
- Li Yao and Ning Zhang, "A Generic Authentication LoA Derivation Model", the 24th IFIP International Information Security Conference (SEC2009), May 18-20, 2009, Coral Beach Hotel, Pafos, Cyprus, IFIP series published by Springer, pp. 98-108

- Nenadic, A.; Ning Zhang; Li Yao; Morrow, T. Levels of Authentication Assurance: an Investigation Information Assurance and Security, 2007. IAS 2007. Third International Symposium on Volume, Issue , 29-31 Aug. 2007 Page(s):155 – 160
- Chin, J.S.; Zhang, N.; Nenadic, A.; Yao, L.; Brooke, J.M.; Towards Context Constrained Authorisation for Pervasive Grid Computing Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on 7-12 Jan. 2007 Page(s):1 – 7
- N. Zhang, L. Yao et al; Achieving Fine-grained Access Control in Virtual Organisations, doi: 10.1002/cpe.v19:9 Concurrency and Computation: Practice and Experience Volume 19 Issue 9, Pages 1333 – 1352, 2006
- N. Zhang, L. Yao, J. Chin, A. Nenadic, A. McNab, A. Rector, C. Goble, Q. Shi, "Plugging a Scalable Authentication Framework into Shibboleth," wetice, pp.271-276, 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), 2005

1.7 Thesis Outline

This thesis identifies and analyses the attributes, processes and procedures that may influence the assurance level of an authentication instance. It identifies and examines four use-case scenarios of authentication in Grid environments and investigates the relationships between the attributes in these scenarios. It then proposes an authentication model, namely a generic e-authentication LoA derivation model (GEA- LoADM). The GEA-LoADM takes account of multiple authentication attributes and their individual LoA values along with their relationships to derive an aggregate LA for the authentication instance. The model has a number of valuable properties, including flexibility and extensibility; it can be applied to different application contexts and supports easy addition of new attributes and removal of obsolete ones. The prototypes of the algorithms and the model have been developed. The performance and security properties of the LoA derivation algorithms and the model are analysed and evaluated against the prototypes. The prototypes are tested against both conventional authentication mechanisms to investigate the performance cost and various attack scenarios to measure the security level. A real-life use case study is presented at the end of the thesis.

In detail, Chapter 2 investigates the related works in the topic area of authentication LoA, provides an in-depth survey of e-authentication technologies and examines the authentication solutions in large-scale distributed system environments. Chapter 3 identifies and investigates authentication LoA-effecting attributes in these environments. It also identifies and analyses the relationships among the attributes and discusses methods to determine the weightings of LoA-effecting attributes. Chapter 4 presents the design, implementation and evaluation of three aggregate authentication LoA derivation algorithms. Chapter 5 describes the design of a generic e-authentication LoA derivation model (GEA-LoADM) including its architecture and architectural components. Chapter 6 prototypes and evaluates the GEA-LoADM model using a real-life case study; the performance and security of the GEA-LoADM model are also analysed. Chapter 7 concludes the thesis and suggests future works.

Chapter 2

Authentication LoA and E-authentication Technologies

2.1 Chapter Introduction

This chapter provides an insight into main-stream authentication technologies and their authentication levels of assurance (LoA). Section 2.2 gives the definition of authentication LoA and discusses related works in defining LoA. Section 2.3 examines main-stream e-authentication technologies and discusses their authentication LoA. Section 2.4 uses the Grid system as an exemplar distributed application to investigate the factors that may have impacts on the authentication LoA. Section 2.5 concludes the chapter and highlights research approaches for the next step.

2.2 Authentication LoA

There have been number of efforts to define and use LoA for transaction or resource protection. These efforts, which will be discussed here, include those by the E-Government/E-Service community [UKloa00, UKloa02, OMB03, Burr06], the Grid/Higher Education (HE) Community [InComm, IGTF-1, IGTF-2], and research communities [Cree04, Conv04, Nena06, Zhan06].

2.2.1 The E-Government Initiative

The first effort to define authentication LoA was made in 2000 by the UK Office of the e–Envoy (now Cabinet Office e-Government Unit) in its 'E-government authentication framework' guideline [UKloa00]. In the guideline, four distinctive authentication assurance levels, ranging from 0 to 3, were identified in terms of the sensitivity and importance of transactions. The document also gives guidance to service providers on how to classify transactions into different groups based on potential impact due to authentication errors, and how to allocate an applicable assurance level to each group. The identified transaction classes and the related authentication assurance levels as

defined in the guidance are summarised below.

- **Level 0**: *Informal transactions.* Misappropriation of identity or repudiation of transaction would not result in inconvenience to the identity holder, risk to their personal safety, financial loss or distress to any party. An authentication service is categorised as Level 0 if no trust is put in the identities claimed by clients.
- **Level 1**: *Personal transactions.* Mistaken identity would have a minor impact to one or more of the involved parties, as information involved is personal but non-sensitive. Misappropriation of identity or repudiation of transaction would not result in major inconvenience to the identity holder, risk to their personal safety, financial loss or distress to any party. For a Level 1 authentication service, users identify themselves by presentation of a credential, which could be a username, and demonstrate the knowledge of a related secret, which could be a password.
- **Level 2**: *Transactions with financial or statutory consequence.* Misappropriation of identity or repudiation of transaction might result in substantial inconvenience to the identity holder (but not risk to their personal safety), significant financial loss or distress to any party. It might also assist in commissioning a serious crime, hinder its detection or materially damage the reputation of the identity holder. For a Level 2 authentication service, users identify themselves by presentation of a credential, which would preferably be a digital certificate, and demonstrate the right to that credential by proving the possession of both the corresponding private key and a password or biometrics. The validity of a credential must be time-bound and the revocation status of the credential must be checked at the time of transaction.
- **Level 3**: *Transactions with substantial financial, statutory or safety consequence.* Misappropriation of identity or repudiation of transaction might result in substantial inconvenience to the identity holder, risk to their personal safety, significant financial loss or distress to any party. For a Level 3 authentication service, users identify themselves by presentation of a digital certificate, which would preferably be stored in a secure hard token, and demonstrate the right to that certificate by proving the possession of both the corresponding private key and a password or biometrics. Face-to-face user registration is required at the time of obtaining a credential, and similar to

Level 2, the validity of the credential must be time-bound and the revocation status of the credential must be checked at the time of transaction.

In September 2002, the third version of this guideline: 'Registration and Authentication e-Government Strategy Framework Policy and Guidelines' [UKloa02], was published. This document builds on the previous Authentication Framework and further specifies that the assurance level in identifying a client should be defined in terms of trust acquired during both the client's *registration* and *authentication* processes. Registration is defined as a complex process consisting of the following steps: identity registration, identity validation, identity verification, credential issuance, logging for audit purposes, and credential withdrawal. Authentication is a process of requesting an identity and verifying it. This document specifies four levels of registration assurance and four levels of authentication assurance, which are appropriate for and can be mapped to the four identified transaction classes. In addition, the document also defines four categories of *identification* with their implied registration and authentication levels as follows:

- **Anonymous or pseudo-anonymous**: neither real-world nor electronic identity is required to complete the transaction (registration level: 0; authentication level: 0).
- **Anonymous or pseudo-anonymous with electronic identity**: the real-world identity of the client is not required to complete the transaction, but the electronic identity enables the service provider to recognise the client in repeated transactions (registration level: 0; authentication level: 1, 2, or 3).
- **Anonymous or pseudo-anonymous with electronic identity and traceable**: the real-world identity of the client is not required to complete the transaction, but the electronic identity enables the service provider to recognise the client in repeated transactions and could be used to trace the real-world identity via the Registration Authority that has registered the client (registration level: 1, 2, or 3; authentication level: 1, 2, or 3).
- **Real-world identity established**: the real-world identity of the client needs to be established to some degree before the transaction can be performed (registration level: 1, 2, or 3; authentication level: 1, 2, or 3).

The UK government's efforts have laid the cornerstone for all subsequent efforts to define and use LoAs. However, these guidelines have only addressed two aspects of LoAs, i.e. registration and authentication. Technical and operational requirements on how to achieve a given level of assurance were missing in the guidelines.

While the UK government guidelines define LoAs in terms of sensitivity levels and importance of transactions, the US Government's Office of Management and Budget (OMB), in its memorandum, 'The E-Authentication Guidance for Federal Agencies' [OMB03], defines levels of authentication assurance in terms of the consequences of the authentication errors and misuse of credentials. This memorandum specifies four assurance levels (Levels 1 to 4), to help and direct US federal agencies in reviewing e-government transactions, determining their authentication needs, and ensuring that the authentication process satisfies the minimum LoA given the risk level measured in terms of potential impacts of authentication errors and the likelihood of their occurrence.

The OMB-defined four assurance levels are as follows:

- **Level 1** - Little or no confidence in the asserted identity's validity
- **Level 2** - Some confidence in the asserted identity's validity
- **Level 3** - High confidence in the asserted identity's validity
- **Level 4** - Very high confidence in the asserted identity's validity

To satisfy each of the assurance levels as specified by the OMB Memorandum, appropriate authentication technologies should be identified, implemented and regularly assessed. The complementary document from the US National Institute of Standards and Technology (NIST), 'NIST SP 800-63: Electronic Authentication Guideline' [Burr06], identifies the necessary technologies and provides guidance to implement the OMB's levels of authentication assurance. The guideline defines specific technical requirements for each of the four levels of assurance. The definition takes into account the effects of authentication token types, authentication protocols and assertion mechanisms for remote communication on the authentication strengths. A short summary of the technical requirements for each of the four levels is provided below.

- **Level 1**: At this level, the authentication mechanism provides some assurance that the same claimant is accessing the protected transaction or data. Successful

authentication requires the claimant to prove that he/she has control of a credential through a secure authentication protocol. Plaintext passwords or secrets are never transmitted across the network. This level does not require the use of cryptographic protocols that block off-line attacks. For example, password challenge-response protocols are allowed at this level, and an eavesdropper, having intercepted such a protocol exchange, can launch an off-line dictionary attack in order to discover the password. Therefore, there is no requirement at this level to use FIPS (Federal Information Processing Standards) approved cryptographic techniques [Fips01]. All the authentication tokens and methods as defined for Level 2, 3, or 4, as well as passwords and PIN, are allowed at this level.

- **Level 2**: This level provides single-factor remote network authentication (SFA). The SFA requires one authentication factor, such as things a user knows (passwords, PINs, shared secrets, solicited personal information, etc). Successful authentication requires the claimant to prove that he/she controls the authentication credential through a secure authentication protocol. The use of cryptographic protocols that can prevent off-line, replay and on-line guessing attacks is required. Any of the authentication tokens of Level 3, or 4 can be used, as well as passwords and PINs.
- **Level 3:** This level provides multi-factor remote network authentication (MFA). The MFA requires more than one authentication factors, such as combining things a user knows and things a user has to form a stronger authentication mechanism. Successful authentication requires the claimant to prove that he/she controls the authentication token (i.e. proves possession of a soft/hardware token) through a secure authentication protocol. Three kinds of tokens may be used: soft cryptographic tokens, hard cryptographic tokens and one-time password device tokens. The use of cryptographic protocols that can prevent off-line replay, on-line guessing, verifier impersonation and man-in-the-middle attacks is required. All sensitive data transferred is cryptographically authenticated and, optionally, encrypted under keys derived from the authentication process. Any of the authentication tokens of Level 4 can be used.
- **Level 4:** This level is intended to provide the highest practical remote network

authentication assurance. Successful authentication requires the claimant to prove that he/she controls the authentication token (i.e. proves possession of a hardware token) through a secure authentication protocol. This level is similar to Level 3, except that only hard cryptographic tokens are allowed. These tokens are hardware (physical) devices that cannot be easily copied and which must be unlocked with a password or a piece of biometric, and thus provide two-factor authentication. Either public or symmetric key technology may be used.

Similar efforts have also been made by the European Government [Euegov], Japanese Government [Jpegov], the Australian Government [Auegov], and the Canadian Government [Caegov] as part of their e-government initiatives. These efforts either use or adopt a similar specification to the OMB/NIST guidance mentioned above. However, these efforts are made in the context of supporting e-Government initiatives. The NIST LoA technical guidance only considers the 'user-to-system' authentication scenario, i.e. covering remote electronic authentication of human users to IT systems via a communication network. The 'system-to-system' and 'credential delegation' scenarios are not covered and discussed (More about these scenarios are discussed in Section 2.4.3). In the 'user-to-system' authentication context, it specifies LoA recommendations versus token/credential types and authentication protocols for registration, identity proofing and records retention.

2.2.2 The Higher Education (HE) and Grid Communities

In addition to the e-Government initiatives mentioned above, a number of HE (Higher Education) federations across the world, including the InCommon federation in USA [Incomm], the Switch (Switzerland) and the AAF (Australia and New Zealand federations [Szegov, Austaf], have defined a number of LoA assurance profiles for the purpose of access management in their respective communities.

The InCommon federation, an access management federation of US HE institutions, uses the Shibboleth authentication and authorization system to enable cost-effective, privacy-preserving and federated identity management among its community of participants. In its draft document, 'Bronze and Silver Credential Assessment Profiles' [Incomm], InCommon recommends two classes of authentication services to be used by the

federation's IdPs and defines two Credential Assessment Profiles (CAPs), namely Bronze and Silver. The Profiles contain the assessment criteria for IdPs wanting to be qualified for providing the Bronze or Silver services. The InCommon Bronze and Silver Profiles only recognise password-based authentication systems that employ Web browsers on the client side. The Profiles neither recognise PKI certificate-based authentication systems, nor systems that use passwords in conjunction with hard (physical) tokens or other specialised hardware or proprietary client software. The reason for this is that campuses across US HE support mainly the use of password-based authentication services, and currently, they do not have any plan for broad adoption of PKI credentials. This is why InCommon have only concentrated on defining profiles compliant with US Gov/NIST Levels 1 and 2 (note that certificate-based authentication is a prerequisite for Levels 3 and 4).

The Swiss HE Federation, SWITCH, has published an informal proposal for using authentication assurance levels on their web site [Szegov]. The Federation funded a pilot project to examine the opportunities and limitations of using LoAs and multi-factor authentication in the Shibboleth infrastructure. The proposal recommends a four-level approach as recognised by the US Government, NIST, E-Authentication and InCommon federations.

Parallel to these activities made in HE sectors, efforts are also being made by the Grid community to improve access control. The most notable efforts in this category have been made by the IGTF (International Grid Trust Federation) WG. IGTF-WG has produced a list of authentication profiles specifying two assurance levels: the Classic X.509 and SLCS (Short Lived Credential) profiles. The Classic X.509 profile, maintained by the European Policy Management Authority for Grid Authentication (EUGridPMA) [IGTF-1], focuses on the operational aspects of CAs (i.e. traditional Certificate); whereas the SLCS profile, maintained by The Americas Grid Policy Management Authority (TAGPMA) [IGTF-2], focuses on the operational aspects of short-lived credential (e.g. proxy certificate) services. The profiles respectively cover traditional and SLCS CAs' operational aspects of LoA, such as identity vetting, certificate expiration, revocation and renewal, the CA's key size, hardware requirements, the physical security of the CA's site and audit and disaster recovery procedures. However, many LoA-effecting factors

brought up by Grid use-case scenarios, such as credential delegation, n-tier authentication, authentication via a third-party and attribute assertion, have not been considered in these profiles.

2.2.3 Other Research Communities

It is worth emphasising that all the LoA guidelines and efforts discussed above centre on the user-to-system authentication use-case scenarios. They do not consider machine-to-machine or software-to-software authentication scenarios. Nor do they address the authentication of a person via a physical authentication mechanism, e.g. location-based or sensor-based services. In addition, issues related to how LoA may be fed into the authorisation process are also outside the scope of these efforts. Most of the existing authentication LoA efforts, such as the one recommended by the OMB/NIST, use an off-line approach to LoA compliance. With this approach, LoA definitions are given as guidelines and the parties concerned are required to comply with these guidelines by conducting a risk assessment of the underlying system, mapping identified risks to an applicable assurance level, selecting appropriate authentication methods and technologies based upon the technical guidelines, and validating the implemented system to make sure that it has achieved the required assurance level. This off-line, or static, approach to authentication assurance level conformance is adequate in a homogeneous environment where resources are also homogeneous; they are well defined, their sensitivity levels are similar and known prior to run-time, and they are often provided by a single service provider, as is the case in e-Government. This approach is certainly not sufficient for large-scale distributed resource sharing environments.

To overcome these limitations, a recent effort has been made to link LoA to authorisation decision-making. The FAME-PERMIS (Flexible Authentication Middleware Extensions to the PERMIS) software derives an LoA value based upon a user's authentication token presented to the authentication service and asserts the value to a role-based access control decision engine run at the SP (Service Provider) side, in order to achieve LoA linked fine-grained access control [Nena06, Zhan06]. However, the software only implements the LoA-definition versus token types as defined by the NIST guideline. It does not consider the impacts of other LoA-effecting attributes. Nor does it provide any algorithms on

deriving authentication LoA given a set of LoA-effecting attributes.

Other related recent works include the algorithms proposed by the ubiquitous computing community [Cree04, Conv04] on the estimation of the trustworthiness of a user. For example, [Cree04] proposes a model to calculate the trustworthiness of a user's pervasive device, and [Conv04] describes a parameterised authentication model for calculating the authentication reliability of authentication sensors in sensor-based networks. These works are largely developed for the context of a ubiquitous computing environment and centred at a broad level of trust, rather than on using authentication LoA to achieve fine-grained access control in large-scale distributed environments such as Data Grids.

2.3 E-Authentication Technologies

2.3.1 Username and Password Authentication

A password is a secret that a claimant memorises and uses to authenticate his/her identity. The proof a user's of knowledge of a password that matches his/her with the username authenticates the user. The username/password-based authentication method is easy to implement and use and is familiar to most users. As a result, it is widely used in e-authentication environments.

With this username/password-based authentication system, the assurance level in identifying a user is mainly dependent on the strength of the chosen password, how it is stored and the underlying authentication protocol used. A successful impersonation of an identity only requires the impersonator to get hold of, or guess, a valid password. Moreover, this method suffers from some password-inherent disadvantages, e.g. difficult-to-guess passwords are difficult to remember, and easy-to-remember passwords are easy to guess. In addition, they are vulnerable to a variety of attacks including guessing, network spoofing and dictionary attacks. Due to these facts, username/password-based authentication method is regarded as providing a rather low authentication level of assurance. NIST e-authentication guideline advises that this method has the authentication LoA of Level 2.

2.3.2 One Time Password Authentication

A one-time password (OTP) is a password that is only valid for a single authentication session or transaction. OTPs avoid many of the shortcomings that are associated with the traditional (static) passwords discussed in the last section. For example, an OTP is difficult to guess since it is generated randomly and users are not required to remember the password. The most important feature of OTP-based authentication is that, in contrast to a static password-based authentication system, it is not vulnerable to replay attacks. This means that, if a potential attacker manages to record an OTP that was previously used to log into a service, he/she will not be able to abuse it since it will be no longer valid. On the downside, an OTP-based authentication system requires the use of additional devices/tokens. In an OTP-based authentication system, the assurance level is mainly dependent on the token that stores the OTPs, how the token is activated (by PIN or biometric) and the underlying authentication protocol used. NIST e-authentication guideline advises that OTP-based authentication systems have the authentication LoA of Level 3.

2.3.3 PKI Software Token Authentication

PKI software token authentication is a multi-factor authentication method, meaning it combines two or more factors to achieve authentication. A PKI software token is a cryptographic key stored in an encrypted file on a host computer (Factor 1: something you have). Typically, the key is encrypted using a secret derived from a pass phase known only to the user (Factor 2: something you know), so the knowledge of the pass phase is required to activate the token. Authentication is accomplished by proving the possession and control of the key. Examples of software tokens include the default Windows Cryptographic Service Provider (CSP) keystore [Marc03] and the Mozilla/Firefox Web browser keystore [Marc03].

Internet Explorer/Windows keystore stores a private key and the certificate of the corresponding public key (which are collectively referred to as a cryptographic credential) as a binary “blob” in the registry by default. This approach makes the private key and certificate as secure as the underlying operating system, and the operating system is

responsible for allowing/denying access to the registry. The Mozilla/Firefox Web browser keystore stores a cryptographic credential in a subdirectory of the application named '*.mozilla*'. The private key of the credential is stored in a file named '*key3.db*' and the certificate(s) in a file named '*cert7.db*' (Firefox uses '*cert8.db*'). Both of these files are binary data in Berkeley DB 1.85 format. These files are protected by the file structure permission and a password specified by the owner of the credential, so that any application capable of reading the Berkeley DB format is still required to provide the password to read/modify the plaintext.

The standard software keystores have the disadvantage of being immobile. Once a private key is installed and used on one host, the only way to transport it to another is to export it onto a portable hard disk or some other mobile storage medium, and re-import it onto the 'new' host. In addition, after exporting the key, the user must remember to erase it from the 'old' host otherwise the key would be at risk of being stolen. As users become more mobile and are expected to use multiple hosts, this immobility hinders its usability. However, as this type of keystore is software-based and the processes that use it are standardised, the costs incurred in its deployment and management are relatively low.

The software token authentication is vulnerable to a number of threats. [Marc03] has given an in-depth investigation of some of these threats. For example, by hijacking a client's CryptoAPI library, '*crypt32.dll*', an attacker can steal the user's private key and use it to forge the client's signatures. For example, an attacker may implant an executable program on the victim's machine and when the victim inputs the password to unlock the key, the program steals the password and acquires a copy of the private key without triggering any extra window that might alert the user. This attack method is particularly useful for stealing medium-or high-security private keys from IE (in a client-side SSL negotiation, for example). Other identified attack techniques include DLL injection and scripts running inside the HTML. Owing to these considerations, the maximum assurance level of this method is defined as Level 3 by NIST e-authentication guidelines.

2.3.4 PKI Hardware Token Authentication

PKI hardware token authentication is also a multi-factor authentication method. A hardware token refers to a cryptographic key that is stored in a hardware device. Access

to the device is typically controlled by a PIN or a user's biometric information. Authentication is accomplished by proving possession of the device as well as controlling the use of the key. Examples of hardware tokens include smart cards and USB tokens. Two API standards have been defined for communicating to a hardware token device in a client desktop environment: the Cryptographic API (CAPI) [CAPIMS] defined by Microsoft for the Windows operating system and the Internet Explorer Web browser, and RSA's PKCS#11 [Pkcs11] that can be used for all operating systems and the Mozilla/Firefox web browser.

When using a hardware token for PKI-based authentication, the key pair is generated inside the card and the public key is exported to a CA for key certification (i.e. the public key is digitally signed by the CA). The private key never comes out of the card, which eliminates the risk of it being stolen or misused. Once the certificate is generated and signed, it is loaded back into the card. Hardware tokens are generally tamper-resistant, and in order to impersonate the card's owner, an attacker must get hold of his token as well as work out the PIN used to lock the card. Hardware tokens are superior to software tokens in terms of keeping private keys secure and supporting users' mobility. However, their deployment can be costly and tedious. Token devices produced by different manufacturers use different drivers and client software, and there is a lack of a standard to unify them and make them inter-operable. For a specific token device connected to a specific operating system, a specific driver and software would have to be installed on each host to be used by a user. This deployment hurdle greatly restricts the wide-scale adoption of this method.

Hardware tokens noticeably increase the security level of the underlying authentication system. According to the NIST Electronic Authentication Guideline, they provide the highest level of assurance (Level 4) among all the known authentication mechanisms. Security threats faced by hardware tokens are of two types: Traditional Mathematical Attacks (TMA) and physical attacks. The TMA attacks are normally done using algorithms that can be modelled and generalised. Examples include the Single-Exponent Multiple-Data attack, the Multiple-Exponent Single-Data attack and the Zero-Exponent Multiple-Data attack [Mess00]. These traditional attacks require the acquisition and manipulation of extremely large amounts of data, and thus may not necessarily be

practical. Physical attacks include Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [Mess00]. With the SPA attack, an attacker uses power consumption patterns to learn the bit values of a secret key. The DPA attack, on the other hand, uses statistical techniques to extract tiny differences in power consumption and extract the bit values of the secret key.

2.3.5 Assertion-based Authentication

Assertions are statements from a verifier or identity provider (IdP) to a service provider that contain authentication information about a claimant. The SP uses the information in the assertion to identify the claimant and make authorisation decisions about his or her access to resources controlled by the SP. An assertion includes identification and authentication statements regarding the claimant, and may additionally include further attribute statements of the claimant to support the authorisation decision at the SP.

Assertion-based authentication supports the process of Single-Sign-On for claimants, allowing them to authenticate once to a verifier and subsequently obtain services from multiple SPs without further authentication. A typical assertion-based authentication process can be broken down into two interactions, one between the claimant and verifier, and the other between the verifier and the SP. The interaction between claimants and verifiers, are similar to the authentication mechanisms described in previous sections, while the interaction between verifiers and SPs can be one of the two assertion modes:

- *Push mode* – In the push mode, a claimant authenticates to a verifier by using an e-authentication mechanism. Following successful authentication of the claimant, the verifier creates an assertion and sends it back to the claimant, which is then forwarded to the SP. The assertion is used by the claimant to authenticate to the SP.
- *Pull mode* – In the pull mode, a claimant authenticates to a verifier by using an e-authentication mechanism. Following successful authentication of the claimant, the verifier creates an assertion as well as an assertion reference (which identifies the verifier and includes a pointer to the full assertion held by the verifier). The assertion reference is sent to the claimant to be forwarded to the SP. In this mode, the assertion reference is used by the claimant to authenticate to the SP. The SP then uses the

assertion reference to explicitly request (pull) the assertion and, if necessary, the attributes of the claimant from the verifier, too.

There are three types of assertion technologies, HTTP cookies, SAML (Security Assertions Markup Language) assertions, and Kerberos.

2.3.5.1 HTTP Cookies

HTTP cookies are one of the most widely used assertion technologies. A HTTP cookie is a small piece of text file used by a browser to store information provided by a particular web site. The content of the cookie is sent back to the web site each time the browser requests a page from it. The web site uses the content to identify the user and prepare customised web pages for that user or to authorise the user for certain transactions.

There are two types of cookies: session cookies, and persistent cookies. A session cookie is erased as soon as the related session is finished, e.g. when a user logs out from an account or when a user closes a web browser. A session cookie is stored in the temporary memory and is not retained after the browser is closed. A persistent cookie, on the other hand, is stored on a user's hard drive until it expires (persistent cookies are set with expiration dates) or until the user deletes it.

Cookies can be used by the claimant to re-authenticate to a server after the communication channel between them has been closed. This may be considered a use of assertion technology in pull mode. In this case, the server acts as a verifier when it sets the cookie in the claimant's browser, and as an SP when it requests the cookie from a claimant who wishes to re-authenticate to it. As shown in Table 2.2, NIST advises an LoA value of 1 to 3 for this technology, depending on the assertion expiration time and assertion integrity protection mechanism used.

2.3.5.2 SAML (Security Assertion Markup Language)

SAML is an XML-based framework for creating and exchanging authentication and attribute-related information between trusted entities over the Internet [Samlv2]. One of the most important design objectives of SAML is to solve the SSO (Single-Sign-On) problem. The building blocks of SAML include the Assertion XML schema, which defines the structure of the assertion; the SAML protocols, which are used to request

assertions and ‘artifacts’ (that is, the assertion reference mentioned in Section 2.3.5); and the bindings that define the underlying communication protocols (such as HTTP or SOAP), which can be used to transport the SAML assertions. SAML supports both the ‘pull’ and ‘push’ modes of assertion. Similar to the Cookie assertion, depending on the assertion expiration time and the assertion integrity protection mechanism used, NIST advises an LoA value of 1 to 3 for this technology. Table 2.1 details the range values.

Cookie/SAML Assertion LoA value	Assertion Integrity Protection	Assertion expiration time
1	Digitally signed/TLS	No requirement
2	Digitally signed/TLS	12 hours
3	Digitally signed/TLS	2 hours

Table 2.1 Cookie and SAML Assertion LoA value range

2.3.5.3 Kerberos

Kerberos [Toda03, Ietf4120] is a network authentication system for use on physically insecure networks. It is based on the symmetric key distribution model proposed by Needham and Schroeder [Need78]. It allows entities communicating over networks to prove their identity to each other while preventing eavesdropping or replay attacks. It also provides for data integrity (detection of modification) and confidentiality (preventing unauthorised reading) using symmetric crypto systems such as DES.

The authentication process proceeds as follows: a claimant sends a request to the verifier (i.e. to an authentication server (AS)) requesting a ‘credential’ for a given SP. The AS responds with the credential encrypted, using the claimant’s master key. The credential consists of 1) a ‘ticket’ for the SP, and 2) a temporary encryption key (often called a “session key”). The claimant forwards (pushes) the ticket (which contains the client’s identity and a copy of the session key, all encrypted in the SP’s master key by the AS) to the SP. The session key (now shared by the claimant and the SP) is used to authenticate the claimant to the SP, and may optionally be used to authenticate the SP to the claimant. It may also be used to encrypt further communications between the two parties or to exchange a separate sub-session key to be used to encrypt further communication.

Depending on the assertion expiration time and the Kerberos ticket generation method, NIST advises assurance values of Level 1 to Level 3, with maximum authentication LoA 3 for this technology. Table 2.2 details the LoA value range.

Kerberos Ticket LoA value	Assertion Integrity Protection	Ticket generation method	Assertion expiration time
1	Digitally signed/TLS	derived from user generated passwords	No requirement
2	Digitally signed/TLS	derived from cryptographic key	12 hours
3	Digitally signed/TLS	derived from cryptographic key	2 hours

Table 2.2 Kerberos Ticket LoA value range

2.3.6 Location-based Authentication

Location-based authentication is an e-authentication technology that proves an individual's identity and authenticity by detecting his/her presence at a distinct location. It can be used to determine whether a person is attempting to log in from an approved location, e.g. a user's office building or home. If any unauthorised activity is detected, it will facilitate finding the individual responsible for that activity.

Compared to conventional authentication technologies, location-based authentication has the benefits of being non-intrusive, i.e. during a location-based authentication process a user is typically not required to explicitly present a credential to a verifier. There are a number of location-based authentication technologies available [Azum93, Hart94, Want92]; however, most of them are designed for ubiquitous computing environments thus providing only local authentication. For example, Active Badge [Want92, Hart94] is a well-known location-based authentication technology. With this technology, an Active Badge is worn on a user and transmits a unique identifier approximately every 10 seconds using an infra-red transmission. Sensors placed at fixed positions within a building receive these infra-red signals and authenticate the user through software. This type of location-based authentication does not provide e-authentication to remote parties unless

there is a way to transmit authentication information from a local sensor to the remote verifier (such as assertion). By doing so, it falls into the category of assertion-based authentication.

2.3.6.1 IP-based Authentication

In IP-based authentication, a verifier identifies a claimant by checking if the IP address of the claimant is on the allowed list defined as part of the verifier's security policy. IP-based authentication requires the confirmation of fixed IP addresses. The method provides a rather low authentication assurance level. It suffers from IP spoofing, and it relies on the user environment to provide user vetting and identification. It is usually used in circumstances that involve many users but where the sensitivity level of the resource does not require individual user identification. An exemplar use of the authentication method is the university online library service, which grants access based upon users' access PC IP addresses within the campus. Therefore, in an authentication environment that requires individual user identification and verification, IP-based authentication can only be used as an auxiliary authentication method in combination with conventional authentication methods to achieve a higher authentication assurance level. The NIST recommendation does not recognise this authentication method, and thus does not recommend any values of authentication LoA.

2.4 Grid Computing Authentication Solutions

In addition to the authentication methods discussed above, there are other factors or attributes that may also have impacts on authentication LoA. This section uses Grid computing, a well known large-scale distributed system, as a platform to identify and investigate the additional LoA-effecting factors and attributes.

2.4.1 Username/Password Authentication Service

Grid Security Infrastructure (GSI) [Welc05], the de facto security solution for Grid, uses the username/password authentication service in conjunction with a Web Service security (WS-Security) framework [Wssv1.1]. WS-Security defines a number of approved token types including Username Token Profile, X.509 Certificate Token Profile, SAML Token Profile and Kerberos Token Profile. Username/password authentication uses the

‘Username Token’ profile. This profile defines how a Web service client can supply a username token as a way of identifying the requester from a username and, optionally, by supplying a password. The following XML snippet shows a sample WS-Security UsernameToken profile:

```
<wsse:UsernameToken wsu:Id="Example">
  <wsse:Username> ... </wsse:Username>
 <wsse:Password Type="..."> ... </wsse:Password>
<wsse:Nonce EncodingType="..."> ... </wsse:Nonce>
  <wsu:Created> ... </wsu:Created>
      </wsse:UsernameToken>
```

Table 2.3 Code snippet of WS-Security UsernameToken

Within the <wsse:UsernameToken> element, a <wsse:Password> element is specified. Two types of passwords are supported: ‘*PasswordText*’ and ‘*PasswordDigest*’. ‘*PasswordText*’ refers to a plain password, whereas ‘*PasswordDigest*’ is the hash of a password. The ‘*PasswordDigest*’ type can only be used if the Plaintext password is available to both the requester and the recipient. Otherwise, the authentication process cannot take place.

When a plain password is used for authentication, the underlying systems must support the use of a secure channel to provide confidentiality and integrity. The transport level security [Tlsv10], or SSL (secure socket layer), is widely used to protect both the token and the entire message body. Message level security [Xmlenc, Xmlsig] may also be used to provide these features for selected parts of a message. If a secure channel is available, the use of digested passwords does not provide an added level of security over the use of plaintext passwords. However, if the underlying transport system does not provide enough protection against eavesdropping, the password should be digested and the ‘*PasswordDigest*’ element should be used. Even so, the password must be strong enough so that simple password guessing attacks will not reveal the secret from a captured message.

When a password is encrypted or hashed, the replay attack threat must be considered. If an attacker can impersonate a user by replaying an encrypted or hashed password, then learning the actual password is not necessary. In WS-Security, the method of preventing a

replay attack is to use a nonce. '<wsse:Nonce>' and '<wsu:Created>' are two optional elements introduced in a 'Username Token' profile for this purpose. A nonce is a random value that the sender creates to be included in each username token that it sends. A SP maintains a cache of used nonces and compares them with new ones to detect replay attacks. A '<wsu:Created>' is a timestamp used to indicate the token's creation time. The advantage of using a creation timestamp together with a nonce is that it allows the server to set an upper-bound time limit to cache the nonce, thus potentially saving the server resources. Also, in order for this mechanism to be effective, the nonce and timestamp must be digitally signed.

This authentication method provides a basic authentication service. Although with the support of a secure communication channel such as an SSL, it also provides confidentiality and message integrity; the password-based authentication method cannot support more advanced security features such as user credential delegation and SSO (Single-Sign-on). Without the support of these security features, complex Grid tasks are difficult to accomplish and/or users may suffer usability problems. For example, a user may need to authenticate himself repeatedly during a job session should the job require services/resources on multiple sites.

As advised in the NIST guidelines, the authentication LoA of this method varies with the types of password used (i.e. complexity of the password) and the underlying system security support. For digest passwords with no secure channel provided, the method has authentication LoA 1. For plain text/digest passwords with the secure channel option, it has authentication LoA 2. Plaintext passwords without the secure channel option stand lower than LoA 1 and are strongly recommended against in real system implementations.

2.4.2 PKI-based Authentication Service

PKI certificate-based authentication is the most commonly used authentication method in Grids. This is not only because it is a well-tested and realisable technology, but also because the trust model of the X.509 ID (identity) credential allows an entity to trust another domain's/organisation's certification authority (CA) without requiring that the rest of its domain/organisation do so. With this method, every Grid user/service is identified by its Distinguish Name (DN) and is issued with a PKI certificate. The

certificate binds the DN of the user/service to its respective public key. In Globus security framework [Welc05], a user may even issue a proxy credential to delegate his/her credential to an entity that may act on his/her behalf, and which can be used for SSO purposes. More about Proxy Certificate will be discussed in Section 2.4.3.

The assurance level of PKI authentication is not only dependent on the strength of the credential, but also on factors related to the length, storage and use of the associated private key. To counter key guessing and brute force attacks, a key pair with sufficient length, and which is created from a true random-number generator, is essential. A viable and secure key storage mechanism is also very important. If an attacker can get hold of another entity's private key, he would then be able to impersonate the legitimate entity and access the resource to which the entity is entitled. Two measures are usually taken to limit the damage caused by private key compromise. The first is to limit the lifetime of a PKI credential (a key pair plus the corresponding certificate). The lifetime of a credential is usually determined by the policy of the CA that issued the certificate. There is obviously a trade-off between security and user convenience in choosing the length of a lifetime for a credential; the longer the lifetime, the higher the risk of a key being compromised, but the less often that the user would have to request a new credential. Typically, the lifetime of a X.509 ID credential is one year, whereas a proxy credential is 24 hours. The second measure focuses on protecting an entity's private key. Apart from software and hardware keystores, an online credential repository (OCR) is another key storage method to achieve key security and user convenience [Novo01]. Depending on the types of the keystore, NIST advises an LoA value of 3 and 4 for this authentication.

2.4.3 Credential Delegation and Single Sign-On (SSO) Solution

Credential delegation and SSO are additional features that are commonly seen in large-scale distributed systems, such as Grid. The following example presents a Grid scenario [Rfc3820] with credential delegation and SSO features and illustrates what a proxy credential is used for in such an authentication process. In a company's Intranet-based Grid, a user, Bob, wants to use a reliable file-transfer service to move a number of large files between various hosts. From his workstation, he submits the transfer requests to a Grid transfer service A. The transfers are expected to take place without Bob's

involvement after the initial request is made. Service A controls the schedule of the transfers. For each transfer, service A will connect to each of the sources and destination hosts and instruct them to initiate a file-transfer connection. A job agent is assumed to be running on Bob's workstation, periodically checking the progress of the transfers by contacting service A. All the transfers and job executions should be performed in a secure manner and Bob uses software token-based PKI credential for authentication in his job execution.

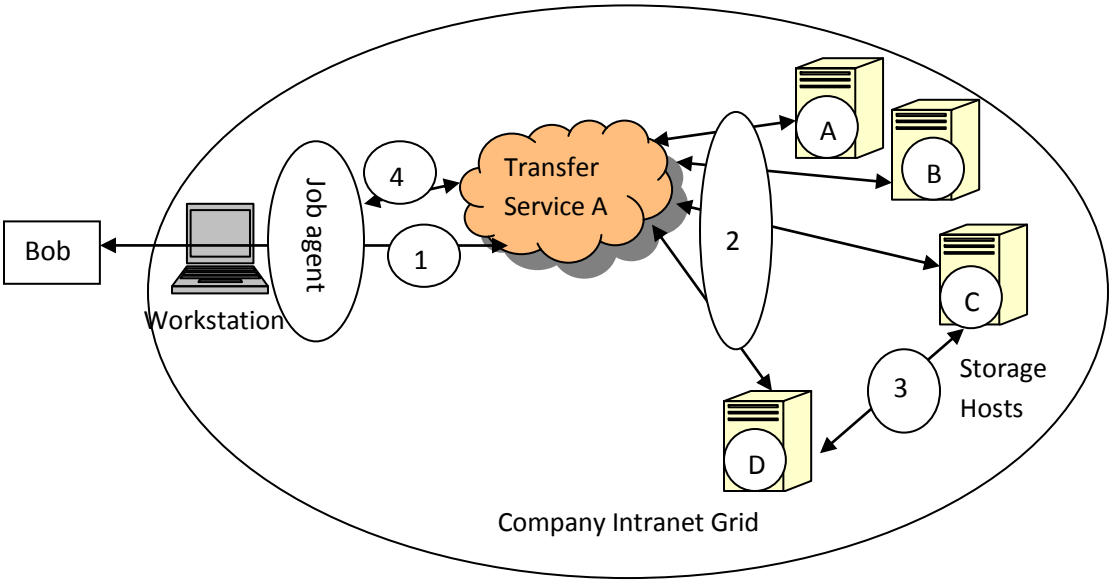

Figure 2.1 A Grid file transfer scenario

As shown in the figure above, this job execution requires authentication in a number of steps. First of all, Bob needs to mutually authenticate with the file-transfer service A to submit the transfer request (step 1). Secondly, service A, with the delegated rights from Bob, should mutually authenticate with the storage hosts A, B, C and D (step 2). Thirdly, the source and destination hosts of a particular transfer must be able to mutually authenticate with each other to ensure the file is being transferred to and from the proper parties (step 3). Finally, the agent running on Bob's workstation must mutually authenticate with the file transfer service in order to check the result of the transfers (step 4).

This scenario sets out two related problems. One is credential delegation and the other is SSO. In this scenario, various remote processes need to perform file-transfer operations

on Bob's behalf, and to ensure these operations are performed in a legitimate manner Bob must delegate the necessary rights to these processes. For example, service A needs to be able to authenticate on Bob's behalf with the source and destination hosts, and Bob must delegate his rights to service A so that it can authenticate with the hosts. It is also convenient for Bob to enter his password once and then run a program that will submit all the file-transfer requests to the transfer service. Bob can then periodically check on the status of the transfers. This program (i.e. the job agent) needs to be given the rights to be able to perform all of these operations securely, without requiring repeated access to the software token or Bob's PIN.

GSI addresses these problems with proxy credentials. A proxy credential consists of a short-term (proxy) private key and a proxy certificate (PC) [Rfc3820]. The PC certifies the corresponding proxy public key. A PC is similar to a X.509 ID certificate, except that the PC is not signed by a CA. Rather, it is signed by its owner with his/her long-term private key corresponding to the public key certified in the ID certificate. A proxy credential is a short-term credential delegated to another entity by its owner and is used in places where the use of the owner's long-term credential is necessary.

By using the proxying technique, Bob's long-term private key stored in the token only needs to be accessed once in order to create a proxy credential. The proxy credential allows the client/agent program to be authorised as Bob when submitting the requests to the transfer service. Access to the long-term credential is not required after the proxy credential creation and within the life time of the proxy credential. The client program on the workstation can delegate to service A the right to act on Bob's behalf. This allows service A to authenticate to the storage hosts on behalf of Bob and to inherit Bob's privileges in order to start the file transfers. Then service A can delegate to the hosts the right to act on Bob's behalf so that each pair of hosts involved in a file-transfer can mutually authenticate to each other in order to perform the file transfer operation in a secure manner. When the agent on the workstation reconnects to service A to check on the status of the transfer operations, it can perform mutual authentication on behalf of Bob.

The Grid community has developed the following versions of PCs with the latest one

fully compliant to the RFC 3820 specification:

- The old version: Legacy proxy certificates used prior to the drafting of RFC 3820. They are deprecated and should only be used when it is necessary to interact with legacy Globus installations. It is recognisable if the 'CommonName' attribute has a value of "proxy" and "limited proxy".
- GT4 Default version: These proxies follow the format defined in RFC 3820, except that they use a proprietary Object ID (1.3.6.1.4.1.3536.1.222) for the 'ProxyCertInfo' extension. It will be replaced with the fully RFC 3820 compliant in a future version of the Globus Toolkit.
- Fully RFC 3820 compliant version: These proxies are fully compliant with RFC 3820. They are expected to be the default in the future.

Compared to a X.509 ID certificate, a proxy certificate has a newly-defined critical X.509 extension, the '*Proxy Certificate Information*' (PCI) extension. The purpose of this extension is to allow the PC issuer to express his/her desire to delegate rights to the PC and also to limit further PCs that can be issued by that PC holder.

There are two fields in the PCI extension. One is the '*PCPathLenConstraint*' field, which specifies the maximum depth of the path of PCs that can be signed with this proxy credential. A zero value of the '*PCPathLenConstraint*' field means that this credential cannot be used to issue other PCs. If the field is not present, then there is no restriction and the maximum proxy path length is unlimited.

Another field is '*proxyPolicy*' which specifies policies on the use of this credential for the purpose of authorisation. There are three modes to delegate issuer's policies through PCs.

- *Full delegation mode* - If an issuer wants to indicate that this is an unrestricted PC that inherits all rights from the issuing credential, he/she can use the '*id-ppl-inheritAll*' value. This mode is in fact an impersonation of its issuer's ID credential. This approach can easily be integrated with a role-based authorisation system since the PC can be treated in the same way as its issuer's certificate. However, from a security point of view, this delegation mode has the highest risk of the credential privileges being misused should the proxy credential be compromised.

- *Restricted delegation mode* - A PC with a restricted usage policy allows an attacker to use it subject to the policy specified in the PCI extension, thus providing more security assurances. However, the implementation of this mode is more complex than the concept. As mentioned before, restricted PCs do not define any particular delegation language. Instead, they provide a structure for issuers to choose a policy language to express their delegation policy. This design increases the flexibility of a PC but causes complications in implementation. The primary issue is that the service provider accepting the restricted PC must understand the delegation language used and be able to enforce the policies that it imposes. Since it is the issuer who chooses the policy language and these policies often contain application-specific restrictions, it is difficult to predict and ensure that all the potential delegating SPs understand the delegation language. According to the Internet X.509 PKI Certificate and Certificate Revocation List (CRL) Profile [Rfc3280], if a critical extension in a certificate cannot be recognised by the SP, then the certificate must be rejected. Therefore SPs that do not understand the delegation language will not be able to accept a restricted PC, even for identity proofing. There have been some attempts to solve the problem. The GSI team tried to extend the API between applications and the security libraries, so that each of them could have the ability to handle and process given restriction delegation policies. For example, the initial implementation of community authorisation service (CAS) [Fost03] uses a restricted PC to delegate the policies. However, this method is difficult in practice because there is no generalised model and it has to be done on a per-application basis. For this reason, the restricted PC has not been widely adopted and implemented.

- *None-Delegation mode* - If an issuer wants to indicate that this is an independent PC that inherits no rights from the issuing certificate, he/she can use the '*id-ppl-independent*' value. In this case, the PC is used solely for the purpose of confirming identity. In this mode, the proxy credential cannot be misused because it does not inherit any rights from its issuer. Thus the security risk is considered low.

A PC issuer can limit what a new PC can be used for by turning off bits in the '*Key Usage*' and '*Extended Key Usage*' extensions. Once a key usage or extended key usage is

removed, the path validation algorithm ensures that it cannot be added back in a subsequent PC. In other words, key usage can only be decreased in a PC chain. The '*PCPathLenConstraint*' field of the '*proxyCertInfo*' extension can be used to limit subsequent delegation of the PC. A service may choose only to authorise a request if it satisfies certain conditions, such as forbidding delegation.

In summary, a proxy credential is generally less secure than an identity X.509 credential. This is because proxy credentials are usually stored in an unencrypted form on a local file system, protected only by the operating system's (OS) file system permissions. A user or a process can access a proxy credential as long as the user logs in to the OS using the account with the access privilege. Therefore, proxy credentials are more vulnerable to theft and abuse. To limit any damage caused by credential compromise or abuse, proxy credentials are typically given much shorter lifetimes than the user's long-term ID credentials (usually in the order of hours or days for proxy credentials versus months or up to a year for ID credentials). Other ways of limiting the damage are to set more restricted policies, limit the key usage and restrict the delegation depth. To sum up, the attributes affecting authentication LoA in proxy credentials are the following:

1) **Proxy credential storage**

   There are basically two methods to store proxy credential file system permission and an online credential repository. File system permission obviously represents a low level of assurance and an online credential repository provides a higher assurance level.

2) **Policy mode**

   As discussed above, there are three policy modes and each policy mode has different privilege and security levels.

3) **Key usage and delegation depth**

   The PC delegation depth is one important attribute to limit the use of the PC. It basically has three conditions; the first is no delegation, the second is one level delegation and the third one is n (n>1) level delegation.

NIST does not identify proxy credential, thus does not advise an authentication LoA for

proxy credential-based authentication.

2.5 The Best Way Forward

Authentication LoA is an important measure of the strength of an e-authentication instance. It can be seen from the discussion above that authentication LoA is dependent on many factors associated with an authentication instance. In order to design a framework that can be used to estimate an aggregate LoA for a given set of LoA-effecting attributes in an authentication instance and to use the estimated LoA to govern access control, we need to 1) investigate all the authentication models and examine their LoA-effecting attributes, 2) study and quantify their respective LoA contributions (called component LoA) to the overall strength of an authentication instance, 3) analyse the attributes' inter-relationships and composite effect on the overall strength (i.e. aggregate LoA or Agg-LoA), and 4) design algorithms to derive the Agg-LoA for the inter-relationship. In this case, if we could classify resources into groups based upon their sensitivity levels and the levels of risks experienced in the underlying access environment; and assign each resource group with a minimum LoA requirement, LoA_R, then we can achieve a more fine-grained access control by linking the Agg-LoA to the LoA_R via run-time authentication method selection and authorisation decision-making. For example, an access request for a particular resource group with the minimum LoA requirement of LoA_R can only be granted if the Agg-LoA derived from an authentication instance is not lower than the minimum LoA value as required by the resource group, i.e. the condition ($\text{Agg-LoA} \geq LoA_R$) must hold. In order to satisfy this condition, the requester would have to choose a stronger form of authentication method that could lead to a sufficient assurance level in identifying the requester. This LoA-linked access control method actually achieves LoA-linked, or in a broad sense, risk-aware access control mapping resource sensitivity levels and/or risk levels in the underlying access environment to the run-time authentication requirement.

Chapter 3

Authentication LoA-Effecting Attributes: Identification and Analysis

3.1 Chapter Introduction

As discussed in Section 2.5, to design the framework that can be used to estimate an aggregate LoA (Agg-LoA) for an authentication instance in an authentication environment and to use the Agg-LoA for fine-grained access control, the following issues should be addressed:

1) Identify all the factors, processes and procedures that may influence the assurance (i.e. confidence) level of the authentication environment; some of these factors will be taken into account as LoA-effecting attributes for LoA derivation in an authentication instance.

2) Quantify or determine a component LoA for each of the LoA-effecting attributes identified in 1).

3) Investigate, identify, and systematically express the inter-relationships among LoA-effecting attributes.

4) Devise algorithms to capture these relationships, and use these algorithms to calculate an Agg-LoA based upon the component LoAs of all the contributing LoA-effecting attributes associated with an authentication process.

This chapter discusses issues 1, 2 and 3. Issue 4 will be addressed in Chapter 4, Section 3.2 presents, in detail, a comprehensive list of LoA-effecting attributes identified by the author, as well as different working groups. Section 3.3 discusses how to quantify component LoA values of individual attributes. Section 3.4 investigates four use-case scenarios in a Grid environment. Based upon these scenarios, two types of relationships among LoA-effecting attributes have been identified; namely the weakest link relationship and the additive relationship. It further investigates potential methods that

can be used to quantify the impact (weighting) of additive attributes on an aggregate LoA.

3.2 Identification of LoA-effecting Attributes

Identification of LoA-effecting attributes has, to some extent, been addressed by existing works, most notably those produced by NIST [Burr06], OASIS (Organisation for the Advancement of Structured Information Standards) [Samlv2], and W3C-WSCWG (W3C web security context working group) [W3cscw].

NIST, in its e-authentication guidance published in 2006 [Burr06], has identified the following LoA-effecting attributes:

- the identity proofing process by which an entity is made known to a registration authority (RA) or a credential service provider (CSP) prior to the issuance and delivery of a credential to the entity;
- the type (and the strength) of the authentication credential used by a remote entity for proving his/her identity, e.g. a PKI credential or a username/password pair;
- the authentication protocol/method used by the underlying authentication service to establish that a remote entity is whom it claims to be;
- the credential management process, which encompasses issues such as the token technology used to store a credential, the manner in which a claimed identity is bound to the credential, the life cycle management of the credentials, and whether the identity provider (IdP) has sufficient operating procedures, processes and policy frameworks to establish the required level of trust;
- whether and how assertion mechanisms are used to convey the outcome of an authentication instance to other parties (e.g. SPs or authorisation engines).

To date, the NIST LoA specification remains the most comprehensive set of guidelines for identity providers implementing systems achieving a pre-defined authentication confidence level. However, the NIST LoA recommendation only covers the user-to-system authentication scenario. It does not address credential delegation scenarios, so it only covers a subset of the LoA-effecting attributes. In addition, with regard to LoA implementation, the NIST work provides guidelines on authentication technologies

versus LoA values which stipulate that the technologies for a given LoA are chosen, implemented and evaluated prior to system run. It has not considered a system that gives users and service providers the run-time choice of using a particular authentication technology, derives an LoA based upon the authentication technology used and the user's authentication environment, and feeds the LoA into the authorisation decision-making process.

The authentication context (AC) specification [Samlac] defined in the SAML v2.0 standard specification set by OASIS has also identified a large number of LoA-effecting attributes (in the OASIS term, they are called Authentication Contexts, or ACs for short), as shown in Table 3.1. The specification has classified the ACs into five categories; *identification*, *technical protection*, *operational protection*, *authentication method* and *governing agreement*. In addition, this specification has also defined an XML-based syntax for constructing ACs. It further uses the syntax to define a number of authentication context classes. Each class contains a subset of the ACs. Classes have been chosen as representative of the current practices and technologies for common authentication, scenarios, such as IP-based authentication, username/password-based authentication etc. The SAML authentication context definition and classification readily allows identity providers and relying parties to negotiate and agree on a mutually acceptable AC.

Category	**ACs**	**Sub-ACs**
Identification:	User identification	• Physical Verification • Written consent • Governing Agreements
Technical Protection	Key store	• Not defined
	Key activation	• Activation PIN
	Key Sharing (Private key protection only)	• Boolean (yes/no)
Authentication	Principal	• Password

Method	Authentication Mechanisms	• Restricted Password • Token • Smartcard • Activation Pin
	Authenticator	• Previous Session • Resume Session • DigSig • Password • Restricted Password • Zero Knowledge • Shared Secret Challenge Response • Shared Secret Dynamic Plaintext • IP Address • Asymmetric Decryption • Asymmetric Key Agreement • Subscriber Line Number • User Suffix • Complex Authenticator
	Authenticator Transport Protocol	• HTTP • SSL • Mobile Network No Encryption • Mobile Network Radio Encryption • Mobile Network EndToEnd Encryption • WTLS • IPSec • PSTN • ISDN • ADSL
Operational Protection	Operational Protection	• Security Audit • Deactivation Call Centre

Governing Agreements	

Table 3.1 List of SAML LoA-effecting attributes (ACs) [Samlac]

The topic of LoA-effecting attributes has been also discussed by the W3C web security context working group (W3C-WSCWG) [W3cscw]. The group's mission is to build consensus around what information people need from browsers in order to understand their security context, to find innovative ways to present this information and raise awareness, and to suggest ways to make browsers less susceptible to spoofing of the user interfaces used to convey critical security information to end users. The W3C-SCWG has specified a baseline set of security context information that should be accessible to web users and defined practices for the secure and usable presentation of this information, thus enabling users to come to a better understanding of the context within which they are operating when making trust decisions on the Web. The work focuses more on educating Web users and on regulating the Web browser to enable a secure and usable interface so that Web users can make safe trust decisions on the Web. The security contexts specified by the group reflects this focus; they are Web-oriented and include Web interactions, user agents, entity identification, third-party recommendation and historical browsing information areas. The limitation of this work is that all the identified security contexts are Web-related. Non-web interaction security contexts are excluded and some important security contexts such as protocols and the user's security environment are not discussed.

Table 3.2 lists a comprehensive set of LoA-effecting attributes by merging and extending the attributes identified in NIST, OASIS and W3C-SCWG's specifications discussed above. Three new attributes have been identified by the author. These newly added attributes (X.509 proxy credential, online credential repository and IP address) are shown in the table in order to differentiate them from work by other people. They are derived from the Grid authentication scenarios to be discussed in Chapter 2.

The attributes listed in Table 3.2 can be classified into two groups: one that cannot be opted by users or SPs at run-time (called Offline LoA-effecting attributes), and the other that can be opted by users or SPs at run-time (called Run-time LoA-effecting attributes). The offline LoA-effecting attributes are typically managed or enforced through governance, rather than run-time choices by the user. An exemplar factor in this group is

user registration procedures, which are usually governed through operational processes and procedures. The run-time LoA-effecting attributes may change dynamically at run-time depending on the authentication or access environment contexts, e.g. the authentication mechanism used and/or the access location of the requester. The values of this group of attributes can be used for deriving the aggregate LoA in a run-time authentication process. N.B. The attributes without asterisks are identified by NIST and SAML, where those with asterisks are identified by the author.

Category	Attributes	Sub-attributes
Offline LoA-effecting attributes	User identification	• Physical • Written content • Online, or close the loop by mail, phone or (possibly) e-mail • Self assertion, minimal records
	Credential Lifetime, Status or Revocation	
	Operational Protection	• Security Audit • Deactivation Call Centre
Run-time LoA-effecting attributes	Authentication Credentials	• Password & PIN • One Time Password • X.509 ID Credential • *X.509 Proxy Credential*
	Authenticator Transport Protocols	• Private key Proof of Possession (PoP) • Symmetric key PoP • Zero knowledge or Tunnelled Password • Challenge-response password
	Key store	• Soft crypto token device • Hard crypto token device • *Online credential repository*
	Assertion	• Digitally signed SAML assertion with 2 hours expiration time • Digitally signed SAML assertion with 12 hours expiration time

		• Digitally signed SAML assertion • Cookie • Kerberos Ticket
	Location*	• * IP-based authentication*

Table 3.2 A comprehensive list of LoA-effecting attributes

Each LoA-effecting attribute has an assurance level, or LoA value. We use the term 'component LoA value' to denote the LoA value of an LoA-effecting attribute. In contrast, we use the term 'aggregate LoA value', to denote an overall LoA value contributed by a set of LoA-effecting attributes. In other words, an aggregate LoA value can be derived from a set of the component LoA values of the attributes involved. The next section focuses on the determination of component LoA values. Issues in relation to the derivation of an aggregate LoA value are to be addressed in subsequent sections.

3.3 Determining Component LoA Values

NIST has defined component LoA values for the attributes it has identified. However, OASIS specification does not define any component LoA for the ACs, nor does it address the impact of the attributes on the aggregate authentication LoA. As a result, we only have a subset of the LoA-effecting attributes that have their component LoA values specified. It is obvious that the task of defining component LoA values should not rest on the author. It should be the responsibility of international communities and standardisation bodies, as it requires the involvement of and consensus between international communities. Based upon this consideration, and the reality that not all of the attributes have their component LoA values defined, we have made an assumption here for our work to proceed without any loss of generality. We assume that all of the attributes that influence an authentication assurance level use the same LoA regime, which could be the NIST LoA regime with LoA values scoped within four levels, Level 1 through to Level 4, or another LoA regime that may emerge in the near future. Working on this assumption, we now focus our efforts on 1) identifying authentication use-case scenarios commonly seen in Grid or other distributed resource sharing environments; 2) analysing the relationships between multiple LoA-effecting attributes in these use-case scenarios; 3) their composite effect on the overall authentication confidence level for a

given authentication instance; and 4) on deriving algorithms that could scientifically and automatically estimate this composite effect (i.e. aggregate LoA) for a given set of LoA-effecting attributes along with their component LoA values.

Table 3.3 shows LoA-effecting attributes versus their exemplar component LoA values. The offline attributes have been omitted since they are not used in run-time aggregate LoA derivation. The values without asterisks are defined by NIST, where those with asterisks have not been defined by international communities but are assigned by the author for illustration purposes.

Category	**Attributes**	**Sub-attributes**	**Maximum Component LoA value**
Run-time LoA-effecting attributes	Authentication Credentials	• Password & PIN	2
		• One Time Password	3
		• X.509 ID Credential	4
		• *X.509 Proxy Credential*	*2*
	Authentication Transport Protocol	• Private key Proof of Procession (PoP)	4
		• Symmetric key PoP	4
		• Zero knowledge or Tunnelled Password	2
		• Challenge-response password	1
	Key store	• Soft crypto token device	3
		• Hard crypto token device	4
		• *Online credential repository*	*3*
	Assertion	• Digitally signed SAML assertion with 2 hours expiration time	3
		• Digitally signed SAML assertion with 12 hours expiration time	2

		• Digitally signed SAML assertion	1
		• Cookie	1
		• Kerberos Ticket	*3*
	Location*	• IP-based authentication	*2*

Table 3.3 LoA-effecting attributes versus their component LoA values

3.4 Identification of Relationships among Multiple LoA-effecting Attributes

To derive an aggregate LoA value from the component LoA values of a set of LoA-effecting attributes (i.e. to analyse and quantify the composite effect of the set of multiple LoA-effecting attributes on the overall confidence level of an authentication instance), we need to be clear about the mutual relationships among the multiple attributes. For this purpose, we have identified and analysed four use-case scenarios of authentication in Grid environments. Figures 3.1 – 3.4 illustrate these use-case scenarios.

- ***Use case 1: user-to-service direct authentication using ID credentials***

In the first use-case scenario, a user authenticates directly to an SP using his/her ID credential. It is also the most generic form of e–authentication in most authentication environments.

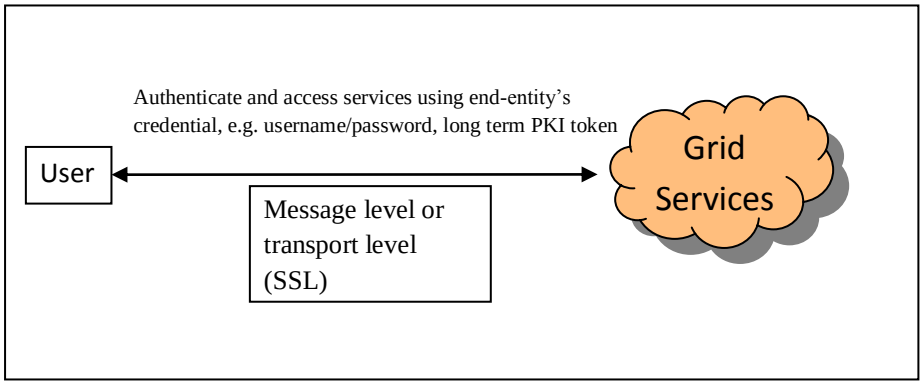

Figure 3.1 Use case 1: user-to-service direct authentication using ID credentials.

- ***Use case 2: user-to-service authentication using proxy credentials stored locally***

In the second use-case scenario, a user identifies her/himself to a local authentication service and upon successful authentication, the user's proxy credential is activated. The

user's proxy credential is then used by the user's client for remote service access. In fact, in this authentication scenario, a single authentication instance consists of two sequential authentication events. The first one is undertaken between the user and the local authentication service. This authentication is typically done through the use of the user's ID credential. The outcome of the authentication is the activation, or issuance, of the user's proxy credential. The second authentication event is between the user's software client to the remote service provider; this is carried out through the use of the proxy credential just activated/issued.

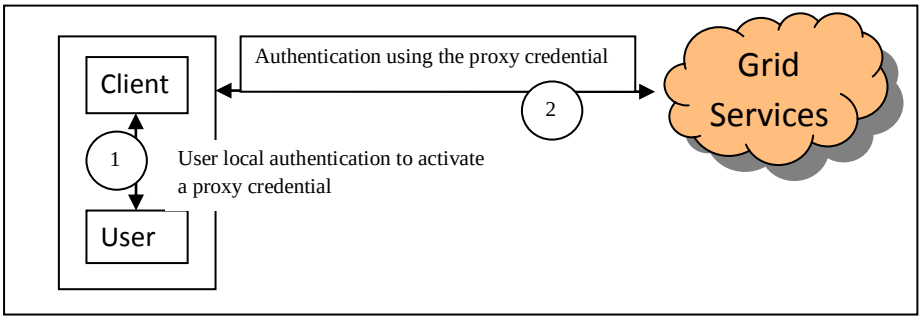

Figure 3.2 Use case 2: user-to-service authentication using proxy credentials stored locally.

- ***Use case 3: user-to-service authentication using proxy credentials stored remotely***

Use-case scenario 3 is similar to use case 2 except that the users' proxy credentials are stored in a remote online central repository (OCR). So, upon successful authentication with the OCR, the user fetches a proxy credential from the repository and then uses it to access remote services; this resembles use case 2. Alternatively, the proxy credential may be dispatched directly from the OCR to the service provider.

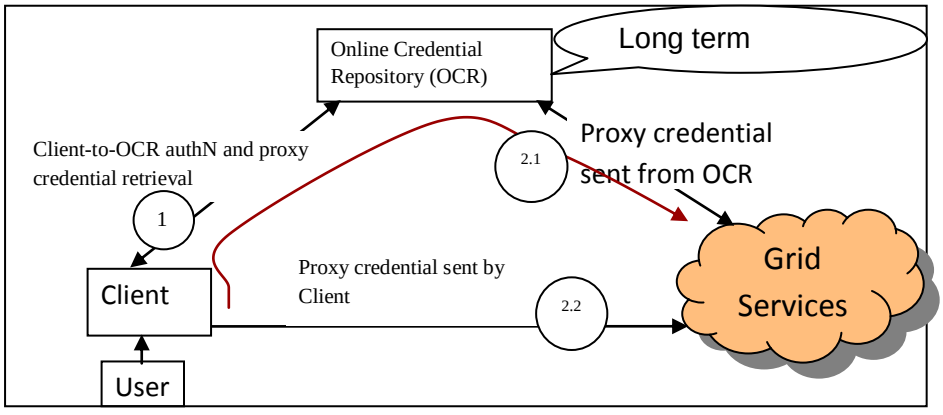

Figure 3.3 Use case 3: user-to-service authentication using proxy credentials stored remotely.

- ***Use case 4: user-to-IdP authentication and IdP asserts the user's identity to remote services (Assertion-based Authentication)***

In the fourth use-case scenario, a user authenticates to a local authentication service (i.e. an identity provider, or IdP) and upon successful authentication, the IdP will assert the user's identity to the remote service provider. Again, a single authentication instance consists of two sequential authentication events. The difference between this use case and use case 2 is that in the second authentication event, instead of using a user's credential to authenticate to the remote service provider, the IdP sends an assertion statement asserting the user's ID or his/her other attributes.

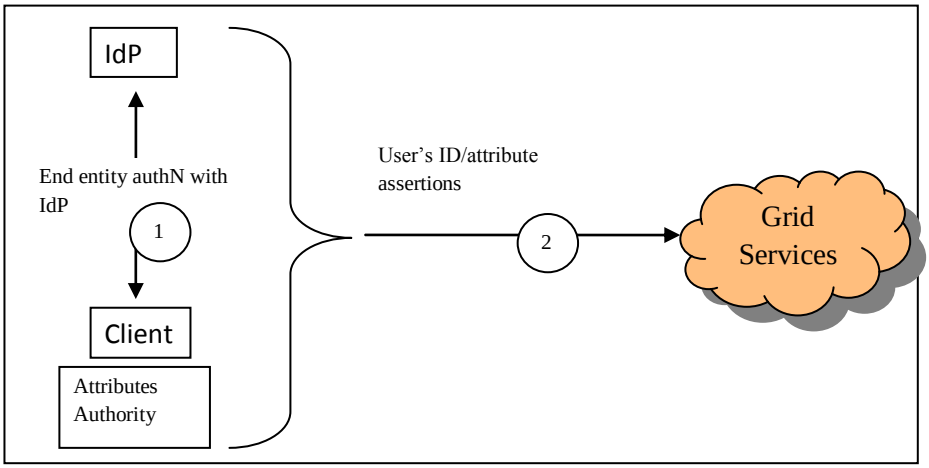

Figure 3.4 Use case 4: user-to-IdP authentication and IdP asserts the user's identity to remote services.

➢ The Weakest-Link Relationship

From these four authentication use-case scenarios, we can make an observation: for some cases, one authentication instance involves a chain of authentication events. If there is one LoA value associated to each of the events, then the overall confidence level (i.e. aggregate LoA) for the entire instance should be equal to the LoA value of the event with the lowest assurance level. In other words, the 'weakest-link principle' applies here, or as the section heading names it 'the weakest-link relationship'. Another exemplar case where the weakest-link principle applies is the relationship between an authentication token attribute and an authentication protocol attribute. A token-based authentication is performed using an authentication protocol; so if a strong token is used in conjunction

with a weak authentication protocol, the aggregate LoA for the authentication instance should not be higher than that of the authentication protocol.

- **The Additive Relationship**

In addition to the weakest-link relationship, the author has identified another important type of relationship among the identified LoA-effecting attributes: the additive relationship. The additive relationship refers to attributes that are independent of, but reinforce, each other so that the aggregate LoA is expected to be at least as strong as the highest component LoA values in the set. For example, if a username and password authentication method is used in conjunction with a location-based authentication method to identify the same user, then the assurance level of this two-factor authentication approach should be higher than the assurance level when either of the services is used alone.

In order to illustrate the different relationships of the attributes accurately to help to design aggregate LoA derivation algorithms, the author has developed a hierarchical structure, namely the LoA-effecting attributes hierarchical structure (LoA-AHS). As shown in Figure 3.5, this structure can accommodate existing-as well as future emerging-LoA-effecting attributes and groups them into different levels and categories. A more detailed discussion is given in Section 5.4.1.1.

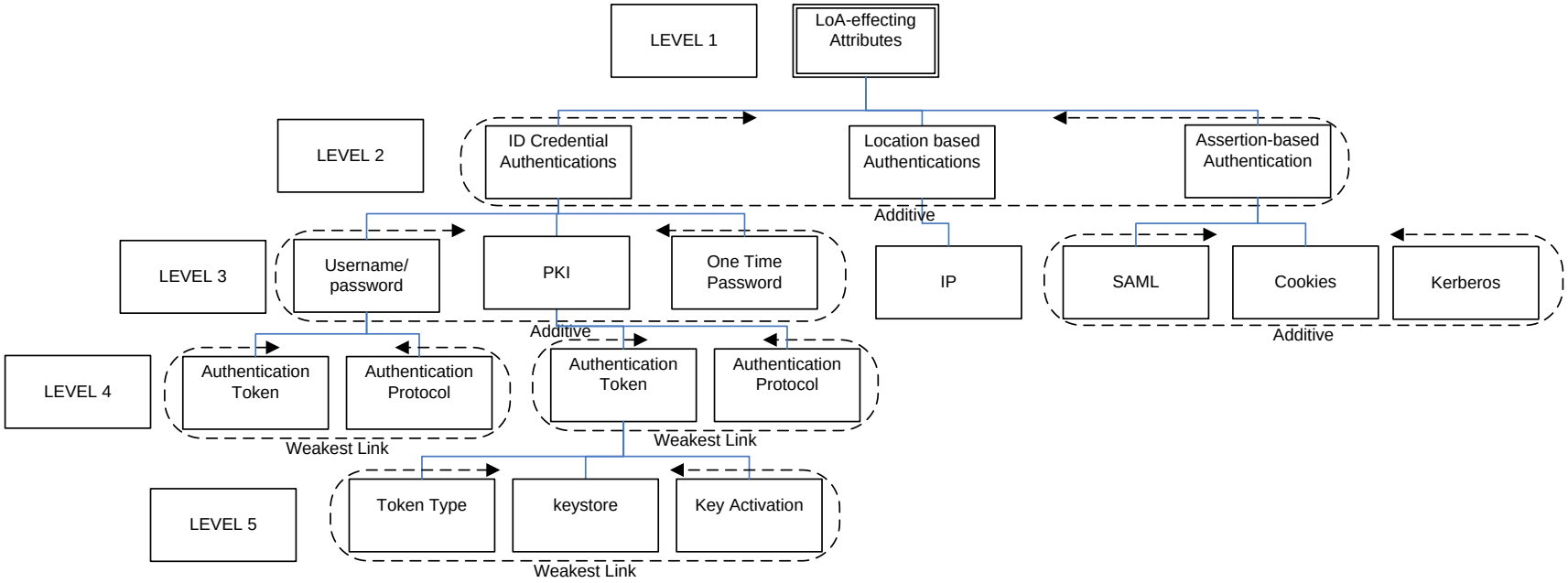

Figure 3.5 Generic LoA-effecting Attributes Hierarchical Structure

When a set of LoA-effecting attributes involved in an authentication instance are in the weakest-link relationship, estimating the aggregate LoA value of the instance is relatively straightforward, the aggregate LoA value is the minimum component LoA value in the set. However, when the set of LoA-effecting attributes are in the additive relationship, the estimation of the aggregate LoA based upon component LoA values of multiple attributes is not that simple. The first hurdle to overcome is how to determine the weightings of the multiple attributes. The next section discusses weighting determination methods.

3.4.1 Determining the Weightings of Additive LoA-effecting attributes

Depending on authentication contexts, environments and access control policies, different additive LoA-effecting attributes may have different levels of impact (i.e. weightings) on the aggregate LoA of an authentication instance. For example, consider the case where a smartcard authentication method attribute with a component LoA value 3 and an IP authentication attribute with a component LoA value 1 are both used in an authentication event. The calculated aggregate LoA values will be significantly different when the ratio of their authentication impact/weighting is (1:1), or when the ratio is (3:1). Therefore, there is a need for a method to calculate the weightings of the attributes. In real life, the quantification and determination of the weightings can be a difficult and subjective process. For instance, for an authentication model using a combined password-based and location-based authentication service, how are the respective impact levels of the password and location attributes determined? The decision may depend on many factors, including the trust levels of the respective authentication services and the underlying access policies defined by the organisation. As the weightings can significantly influence the aggregate LoA of an authentication instance, an interesting research investigation is to search for a method that can scientifically determine the weightings of multiple LoA-effecting attributes for a given authentication process and make the method adaptable to the variations in the LoA-effecting attribute types used. We survey the multi-criteria decision making (MCDM) techniques and explore the use of a MCDM technique-the Analytic Hierarchy Process (AHP)-to address this research issue.

3.4.1.1 Three Multi-Criteria Decision Making (MCDM) Approaches

MCDM (also referred to as Multi-Criteria Decision Analysis) is defined by the International Society on Multiple Criteria Decision Making [Ismcdm] as the study of methods and procedures by which concerns about multiple criteria (attributes) in different aspects can be formally and systemically incorporated into a decision-making process.

The weighting determination of additive LoA-effecting attributes can, in a sense, be seen as a multi-criteria (i.e. multiple LoA-effecting attributes) decision-making problem (i.e. weight determination). An MCDM method uses a decision matrix of criteria and performance scores to provide a systematic analytical approach for integrating risk levels, uncertainty criteria/attributes, and valuation to enable the evaluation and ranking of many alternatives (i.e. the possible decisions that are required to select). Almost all of the MCDM methods share a similar approach to criteria organisation and decision matrix construction, but differ in that they synthesise information differently [Yoec02]. They use different types of value information and different optimisation algorithms. For example, some methods rank options, some identify a single optimal alternative and some provide an incomplete ranking, while others differentiate among acceptable and unacceptable alternatives.

The different MCDM methods [Royb05] can generally be classified into three approaches: the multi-attribute preference theory approach, the analytic hierarchy process (AHP), and the outranking approach, though the AHP approach can be viewed as a special case or an approximation of a multi-attribute preference model [Royb05 chapter 7, Chae04].

The multi-attribute preference theory approach [Keen93] consists of several multi-attribute models. These models are based on alternate sets of axioms that have implications for their assessment and use. Keeney and Raiffain [Keen93] differentiate between these models based on the notions of (a) ordinal comparisons and (b) strength of preference versus theories for risky choices. The term ‘multi-attribute value function’ is used to refer to the former (i.e. case a), and multi-attribute utility function (MAUT) is used to refer to the latter (case b). The goal of MAUT is to find a simple expression of the net benefits of a decision. Through the use of a utility or value function, MAUT transforms diverse criteria into one common scale of utility or value. MAUT relies on the

assumptions that the decision-maker is rational (e.g. preferring more utilities to fewer utilities), has perfect knowledge and is consistent in his judgments [Kike05]. The goal of the decision-maker in this process is to maximise the utility or value. Because poor scores on certain criteria can be compensated for by high scores on other criteria, MAUT is part of a group of MCDA techniques known as 'compensatory' methods.

Similar to MAUT, AHP [Saat90] aggregates various facets of the decision problem using a single optimisation function known as an objective function. The goal of AHP is to select an alternative that results in the greatest value of the objective function. AHP uses a quantitative comparison method that is based on pair-wise comparisons of decision criteria/attributes. All individual criteria/attributes must be paired against all others and the results compiled in a matrix. The user uses a numerical scale (1 to 9) to compare the criteria/attributes and AHP moves systematically through all of the pair-wise comparisons of criteria/attributes and alternatives. AHP thus relies on the supposition that humans are more capable of making relative judgments than absolute judgments [Royb05 Chapter 9].

Another approach, the outranking approach, is based on the principle that one alternative may have a degree of dominance over another [Kang01]. Dominance occurs when one alternative performs better than another on at least one criterion, and no worse than the other on all criteria [Odpm04]. However, an outranking technique does not presume that a single best alternative can be identified. It compares the performance of two (or more) alternatives at a time, initially in terms of each criterion, to identify the extent to which a preference for one over the other can be declared. It then aggregates the preference information across all relevant criteria and attempts to establish the strength of the evidence to favour the selection of one alternative over the others. For example, an outranking technique may be in favour of an alternative that performs the best on the greatest number of criteria. Thus, outranking techniques allow inferior performance on some criteria to be compensated for by superior performance on others.

3.4.1.2 Choosing the Most Suitable Approach

Table 3.4 summaries the strengths and weaknesses of the three MCDM approaches discussed above.

The characteristic of the outranking approach requires the comparisons of alternatives

against criteria. However our problem is not to select alternatives based on multiple criteria, but rather to systematically compute the weightings of a set of contributing LoA-effecting criteria/attributes. There are no alternatives in our problem, so the outranking approach is not suitable for the weighting calculation.

In MAUT, criteria weightings are often obtained by directly surveying stakeholders, as shown in table 3.4. Rigorous stakeholder preference solicitations are expensive, and less rigorous stakeholders' surveys may not accurately reflect stakeholders' true preferences. This stakeholder-based approach to criteria weightings does not address our problem effectively as this weighing assignment process is entirely subjective, leading to a lack of commonality, and is problematic for the implementation of interoperability. For this reason, the MAUT approach is ruled out.

AHP, on the other hand, uses a pair-wise comparison to calculate the criteria weightings. This weighting calculation method is systematic, flexible and relatively inexpensive in comparison to the stakeholder-based approach. It has been proven to be useful by both theoretical and practical applications in the business arena [Korp04, Gass05, Saat90]. Most importantly, because it uses a hierarchical structure in its solution that matches the LoA-effecting attributes structure detailed in Section 3.4, it reflects how the LoA-effecting attribute weightings should be calculated. Therefore, the next part of this section will focus on the discussion of AHP procedures and their suitability for solving our problem.

Method	Important elements	Strengths	Weaknesses
MAUT	• Expression of overall performance of an alternative in a single, non-monetary number representing the utility of that alternative • Criteria weights often obtained by directly surveying stakeholders	• Easier to compare alternatives whose overall scores are expressed as single numbers • Choice of an alternative can be transparent if highest scoring alternative is chosen	• Maximisation of utility may not be important to decision makers • Criteria weights obtained through less rigorous stakeholder surveys may not accurately reflect stakeholders' true preferences • Rigorous stakeholder preference elicitations are expensive
AHP	• Criteria weights and scores are based on pair-wise comparisons of criteria and alternatives, respectively	• Theoretically sound–based on utilitarian philosophy • Many people prefer to express net utility in non-monetary terms • Surveying pair-wise comparisons is easy to implement	• The weights obtained from pair-wise comparison are criticised for not reflecting people's true references • Mathematical procedures can yield illogical results. For example, rankings developed through AHP are sometimes not transitive
Outranking	• One option outranks another if: (1) it outperforms the other on enough criteria of sufficient importance (as reflected by the sum of criteria weights), and (2) it is not outperformed by the others, in the sense of recording a significantly inferior performance on any one criterion. Does not require the reduction of all criteria to a single unit Does not always take into account whether over-performance on one criterion can make up for underperformance on another • Allows options to be classified as "incomparable"	• Does not require the reduction of all criteria to a single unit • Explicit consideration of possibility that very poor performance on a single criterion may eliminate an alternative from consideration, even if that criterion's performance is compensated for by very good performance on other criteria	• Does not always take into account whether over- performance on one criterion can make up for underperformance on another • The algorithms used in outranking are often relatively complex and not well understood by decision-makers

Table 3.4 Comparison of critical elements, strengths and weaknesses of three advanced MCDA methods: MAUT, AHP, and Outranking [Kike05]

3.1.4.3 Analytic Hierarchy Process (AHP)

AHP structures multiple attributes into a hierarchy, assesses the relative importance of these attributes, compares alternatives (possible decisions) for each attribute, and finally, determines an overall ranking of the alternatives [Saat90]. We use this AHP method to assess the relative importance of multiple attributes to determine the weightings of the additive LoA-effecting attributes used in an authentication environment.

According to [Saat90], five steps are involved when using AHP method to solve a multi-attribute decision problem, and the following four steps are used for calculating the weightings of a set of additive LoA-effecting attributes.

- **Step 1. Structure an AHP structure from the decision problem**

AHP decomposes a decision problem (which is not structured at this stage) into decision elements and organises them into a hierarchical structure. Navigating through the hierarchy from top to bottom, the AHP hierarchical structure comprises a goal, one or more criteria (evaluation parameters, i.e. attributes) and a set of alternatives (possible decisions). Building this hierarchy requires a thorough analysis of the problem. This involves, firstly, the identification of the attributes that either directly or potentially affects the problem and, secondly, the identification of possible alternatives associated with the problem. Figure 3.6 shows an exemplar LoA-AHS structure. It fits the AHP requirement perfectly.

The benefits for arranging the goal, attributes and alternatives into a hierarchical structure are twofold. Firstly, it provides an overall view of the complex relationships among the attributes. Secondly, it helps decision-makers to assess whether issues at each level are of the same order of magnitude and to compare any homogeneous elements more accurately.

Decision-makers take full control of the hierarchy. They may insert or eliminate levels and/or attributes as necessary to clarify priorities or to sharpen the focus on

one or more aspects of the problem. Attributes that have a global character can be represented at a higher level of the hierarchy while others, which specifically characterise the problem, can be represented at a lower level.

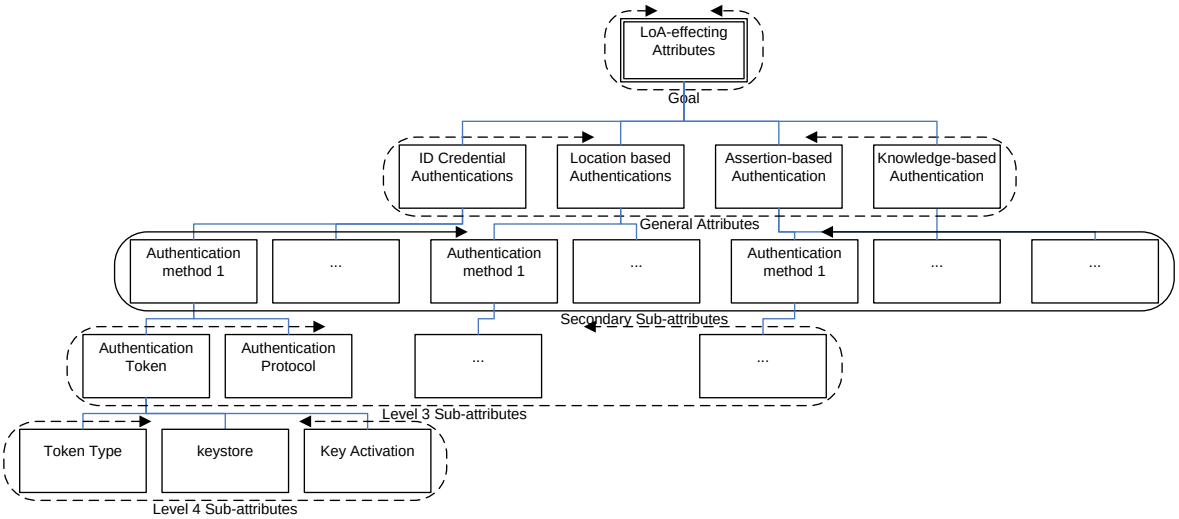

Figure 3.6 An exemplar LoA-AHS structure

- **Step 2. Create the input values by pair-wise comparisons of decision elements**

In order to quantitatively measure the impact of each attribute on the goal, AHP derives relative scales using judgments or data from a standard scale and performs an arithmetic operation on the scales. The judgments are given in the form of pair-wise comparisons [Saat77, Saat80], and the standard scale is developed as the fundamental scale [Saat82].

One of the benefits of this hierarchical approach is that it allows us to focus on each of the multiple properties essential for making a sound decision separately and respectively. We can, in turn, take a pair of elements and compare them on a single property without worrying about other properties or other elements. This is why combining pair-wise comparisons with the hierarchical structure approach is a useful methodology in decision-making [Saat90].

The fundamental scale, shown in Table 3.5 and developed by Prof. Saaty [Saat77],

is used to represent the intensity of importance among the attributes in question. The effectiveness and theoretical justification of the fundamental scale have been validated by [Royb05 Chapter 9]. The scale has nine levels, from 1 through 9. Five of them (1, 3, 5, 7, 9) represent equal, moderately important, strongly important, very strongly important, and extremely important, respectively. The other four (2, 4, 6, 8) refine the scales by taking the median values of 1, 3, 5, 7, and 9, respectively.

Intensity of importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the object
2	Weak	Between Equal and Moderate
3	Moderate importance	Experience and judgment slightly favour one activity over another
4	Moderate plus	Between Moderate and Strong
5	Strong importance	Experience and judgment strongly favour one activity over another
6	Strong plus	Between Strong and Very Strong
7	Very strong or demonstrated importance	An activity is favoured very strongly over another; its dominance is demonstrated in practice
8	Very, very strong	Between Very Strong and Extreme
9	Extreme importance	The evidence favouring one activity over another is compelling

Table 3.5 The Fundamental Scale for AHP [Royb05]

The pair-wise comparison can be expressed using a matrix. Assume that we have a problem with *n* attributes, {a_1, a_2, … , a_n}. Element a_{ij} is used to denote the intensity of importance (Table 3.5) difference between two attributes, a_i and a_j (i.e. a_i/a_j). Thus, we have the comparison matrix,

$$A=\left(a_{ij}\right)_{n\times n}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}=\begin{pmatrix} 1 & a_{12} & \cdots & a_{1n} \\ 1/a_{12} & 1 & \cdots & a_{2n} \\ \cdots & \cdots & 1 & \cdots \\ 1/a_{1n} & 1/a_{2n} & \cdots & 1 \end{pmatrix} \qquad (3.1)$$

where $a_{ii}=1, a_{ij}=\dfrac{1}{a_{ji}}, a_{ij}>0, \forall i=1,2,...,n$.

From matrix ***A***, it can be seen that with n*n elements in a matrix, one needs $(n^2-n)/2$ comparisons. This is because there are n 1's on the diagonal for comparing elements with themselves, and of the remaining judgments, half are reciprocals.

- **Step 3. Estimate the relative weightings of the attributes**

Once the comparison matrix is constructed, the weightings of elements (attributes) $a_i, \forall i=1,2,...,n$ can be determined. Assume w_i is the weight of attribute a_i and $W=[w_1, w_2, w_3,...w_n]$ denotes the set of weightings for *n* attributes { $a_i, \forall i=1,2,...,n$ }, respectively, then $a_{mn}=w_m/w_n, \forall m,n=1,2,...,n$. Using the AHP method, *W* can be calculated from matrix A by recovering the vector $W=[w_1, w_2, w_3,...w_n]$ as follows,

$$AW = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{pmatrix} = \tag{3.2}$$

$$\begin{pmatrix} w_1/w_1 & w_1/w_2 & \cdots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \cdots & w_2/w_n \\ \cdots & \cdots & & \cdots \\ w_n/w_1 & w_n/w_2 & \cdots & w_n/w_n \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{pmatrix} = n \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{pmatrix} = nW$$

The equation multiplies A on the right by the vector of weightings, $W = [w_1, w_2, w_3, ...w_n]$. The result of this multiplication is nW. [Saat90, Royb05 Chapter 9] have proved that n is an eigenvalue of ***A***, and W is the corresponding eigenvector of n. The solution of AW = nW, called the principal right eigenvector of ***A***.

- **Step 4. Check for result consistency**

According to [Saat90], once the weightings of the matrix are determined, the consistency of the matrix should be checked before the weightings can be trusted. [Saat90] also proved that a comparison matrix A is said to be consistent if and only if $a_{ij}a_{jk} = a_{ik}$ for all i, j and k.

It has been proved in [Saat90] that for consistent reciprocal matrices (equation 3.1), the largest eigenvalue $\lambda_{\max}$ is equal to the number of comparisons, i.e. $\lambda_{\max} = n$. Based on this, a measurement of consistency is given, which is called the Consistency Index (CI) and is defined as the deviation or degree of consistency using the following equation:

$$CI = \frac{\lambda_{\max} - n}{n-1} \tag{3.3}$$

In order to find out what the CI might mean, a simulation solution has been developed by [Royb05, chapter9 pp374-375]. In the simulation, choosing the entries of ***A*** above the main diagonal at random from the 17 values {1/9, 1/8, ⋯ , 1, 2, ⋯ ,

8, 9}. Then fill in the entries of *A* below the diagonal by taking reciprocals. Put ones down the main diagonal and compute the consistency index. Do this many thousands of times and take the average, which they call it the random consistency index (RI) as shown in Table 3.6. The CI is then divided by the RI to calculate the consistency ratio (CR), i.e.

$$CR = CI/RI \qquad (3.4)$$

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.52	0.89	1.11	1.25	1.35	1.40	1.45	1.49

Table 3.6 Random Consistency Index values

The following rules are used to judge whether or not matrix A is consistent, please note that the 0.1 ratio threshold is carefully specified by the [Saat90]:

- If CR = 0, then matrix ***A*** is consistent;
- If CR $\leq$0.10, then matrix ***A*** is considered as acceptably consistent;
- If CR > 0.10, then matrix ***A*** is not consistent and the subjective judgements need to be revised. I.e. repeat Step 2 to Step 4 until the matrix ***A*** is at least acceptably consistent.

Our required algorithm needs to be able to derive an aggregate authentication LoA value based on the component LoAs of contributing LoA-effecting attributes associated to an authentication process, along with the impacts of these attributes. The AHP method is used here to analyse and quantify the respective impacts (i.e. weights) of the LoA-effecting attributes and to derive an aggregate LoA value, taking into account the impacts and component LoA values of these multiple attributes. The application of this AHP method to the determination of the weightings of multiple additive LoA-effecting attributes is illustrated in Section 4.4.1.

3.5 Chapter Conclusion

In this chapter, we have investigated and analysed component LoA values associated with a comprehensive set of LoA-effecting attributes. We then identified two types of relationships among those attributes. In addition, we surveyed and chose an MCDM method to quantify the weightings of the additive attributes. These are all essential steps prior to the design of aggregate LoA derivation algorithms. In the next chapter, we will present the design of the algorithms by integrating the LoA-effecting attributes identified in this chapter with their component LoA values and by taking into consideration the relationships and weightings of the attributes.

Chapter 4

Authentication LoA Derivation Algorithms: Design, Implementation and Evaluation

4.1 Chapter Introduction

As discussed in Section 3.4, for any given authentication instance there will be a set of multiple LoA-effecting attributes, which can be organised into an LoA-AHS structure. Using the structure, we can estimate the composite effect on the assurance level in identifying a user (i.e. aggregate LoA) of these attributes. This is done in a bottom-up manner, assuming that there are m levels (levels 1, ..., m) in the structure. From the bottom level m, based on the relationship (the weakest link, or the additive) of the group of attributes at the level, an aggregate LoA derivation algorithm (corresponding to the relationship) is used to calculate an aggregate LoA for the attribute group at this level. This aggregate LoA value is then used as the component LoA of the connected attribute of the level immediately above, i.e. Level (m-1). This process continues until the top level, i.e. Level 1, of the structure is reached and the aggregate LoA value at Level 1 is the overall confidence level, i.e. the aggregate LoA, for the entire authentication instance.

Obviously, for different relationships among multiple attributes, different LoA derivation algorithms should be used. This chapter presents the design and evaluation of the algorithms. Section 4.2 specifies the design requirements for the algorithms. Section 4.3 and 4.4 describe the weakest-link relationship Agg-LoA derivation ($ALoA_{WL}$) algorithm and additive relationship Agg-LoA derivation ($ALoA_{AD}$) algorithm, respectively. Section 4.5 presents a performance evaluation and an algorithm result evaluation. Section 4.6 summaries the chapter.

4.2 Design Requirements

This section specifies three requirements (R1, R2, R3) for the design of the algorithms.

R1 – Derivation of aggregate LoA:

Given a set of LoA-effecting attributes, their component LoA values, their respective weightings and their mutual relationship, there is an algorithm that should be able to derive an aggregate LoA value for this set of attributes.

R2 – Algorithm efficiency:

Computational and communication overheads introduced as a result of using the algorithms should be kept as low as possible.

R3 – Support the use of policy-driven attribute LoAs:

Each of the LoA-effecting attributes will have a component LoA associated with it. The values of these component LoAs are usually dependent on the LoA regime used, which should be defined by relevant international communities. To support the use of any-either existing (such as the NIST definition [Burr06]) or emerging-LoA regimes, the algorithms will use a policy-driven component to handle component LoA values.

Each set imposes an aggregate LoA value on an attribute at the level immediately above this level. If multiple attributes in a set are in the weakest-link relationship, then the $ALoA_{WL}$ algorithm should be used to calculate the aggregate LoA for this set. This aggregate LoA value is then taken as the component LoA value for the attribute at the level above this set. Similarly, if a set's attributes are in an additive relationship then the aggregate LoA for this set should be derived using the $ALoA_{AD}$ algorithm.

4.3 Aggregate LoA Derivation for Attributes in the Weakest Link Relationship –the $ALoA_{WL}$ Algorithm

Assume that at level *k* there are multiple sets of attributes $\{SET_A, SET_B, \ldots, SET_N\}$. Also assume that the attributes $\{a_1, a_2, \ldots, a_n\}$ in SET_A are in the weakest-link relationship, and their respective component LoA values are $\{LoA_{a1}, LoA_{a2}, \ldots, LoA_{an}\}$. Then the composite effect of these attributes in SET_A on the authentication assurance level of the attribute at the level immediately above should be the lowest component LoA value in the set. Mathematically, this can be expressed as:

$$Agg\_LoA_{(WL,\ SET\_N,\ level\_k)} = min(LoA_{a1}, LoA_{a2}, \ldots, LoA_{an}) \qquad (4.1)$$

where *min* is the minimum function, and $Agg\_LoA_{(WL,\ SET\_N,\ level\_k)}$ is the aggregate LoA value for set N at level k.

From this discussion, it can be seen that the derivation of an aggregate LoA value for a group of attributes that are in the weakest-link relationship only requires the attributes' component LoA values. The pseudo code for the algorithm is given below:

```
INPUT: LoA-effecting attributes Set N, N>1
OUTPUT: Agg-LoA Value Agg-LoA
Integer Method Weakest_LoA(Object Set N)
   WHILE Set N STILL HAS ELEMENT
      READ one attribute LoA value in N;
         SAVE into Integer ARRAY LoA[];
   ENDWHILE
   SORT ARRAY LoA[];
   Agg-LoA = the Lowest LoA[x];
   RETURN Agg-LoA;
```

Table 4.1 The weakest link algorithm pseudo code

An implementation of this algorithm is in Appendix A and the evaluation of this algorithm is undertaken in Section 4.5.

4.4 Aggregate LoA Derivation for Attributes in the Additive Relationship –the $ALoA_{AD}$ Algorithms

An $ALoA_{AD}$ algorithm is required for estimating an aggregate LoA value given a set of attributes that are in an additive relationship. Three tasks are required to devise the algorithm. The first is to determine the attributes' weightings. This can be done by using the AHP technique, as discussed in Section 3.4.1.3. The second task is to find a way to integrate the weightings into the algorithm. The third task is to find a method to quantify the reinforcement effect collectively contributed by the additive attributes. Section 4.4.1 describes a method for the integration of the weightings of each attribute into their corresponding component LoA values. Section 4.4.2 and 4.4.3 present two $ALoA_{AD}$ algorithms that could be used to derive an aggregate LoA value given component LoA values of a set of additive LoA-effecting attributes.

4.4.1 Attribute Weightings and Their Integration with Component LoA Values

In Section 3.4.1, the author pointed out that when a set of attributes are in an additive relationship, the quantification of their composite effect on the assurance level of an authentication instance requires the consideration of their respective weightings. Also in that section, the author proposed to use the AHP technique to calculate the weightings for a set of additive LoA-effecting attributes. Here in this section, we demonstrate how attributes' weightings are determined and how these weightings are integrated into the aggregate LoA derivation process.

Figure 4.1 shows an exemplar LoA-AHS structure in an authentication environment. Level-2 in Figure 4.1 shows a set of three additive attributes, Set A = {'ID Credential-based Authentication', 'Location-based Authentication', 'Assertion-

based Authentication'}. Using the AHP method described in Section 3.4.1, the weightings of these three attributes are determined as follows.

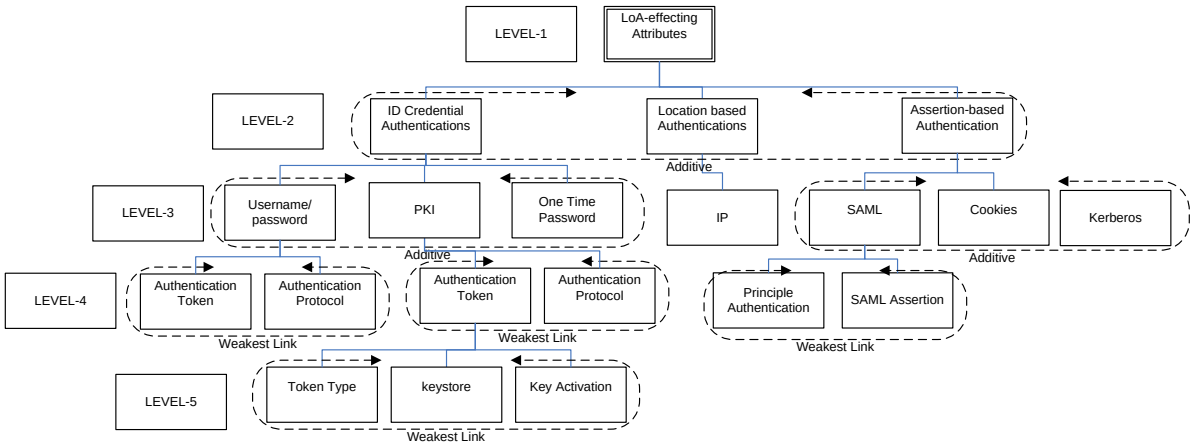

Figure 4.1 An exemplar LoA-AHS structure in an authentication environment

1) Based on the fundamental scale as shown in Table 3.5, a decision-maker (e.g. an administrator managing the authentication environment) uses his/her subjective opinion and inputs pair-wise comparison values (as shown in Table 4.2) and constructs matrix $A = \begin{bmatrix} 1 & 6 & 3 \\ \frac{1}{6} & 1 & 4 \\ \frac{1}{3} & \frac{1}{4} & 1 \end{bmatrix}$.

	ID credential-based Authentication	Location-based Authentication	Assertion-based Authentication
ID credential-based Authentication	1	6/1	3/1
Location-based Authentication	1/6	1	1/4
Assertion-based Authentication	1/3	4	1

Table 4.2 Pair-wise comparison matrix for the attribute set

2) Compute matrix A's principle eigenvalue $\lambda_{\max} = 3.054$ and the corresponding

eigenvector $W = [0.915, 0.121, 0.384]$ using the equation (3.2).

3) Check for consistency by calculating the consistency ratio (CR), and if the CR is equal to 0 or less than 0.1, the matrix is consistent. In this case, by using the equation (3.3) we can calculate CI=0.027, and because the size of matrix *A* is 3, thus RI=0.52, thus CR=CI/RI=0.052<0.1, which indicates that matrix *A* is consistent.

4) If matrix *A* is consistent or acceptably consistent, the normalised eigenvector $W' = [0.644, 0.085, 0.271]$ is derived from W, and W' is the normalised weight (i.e. weights summing to one) for the attribute set {'ID Credential-based Authentication', 'Location-based Authentication', 'Assertion-based Authentication'}.

5) Repeat steps (1)-(4) above for every additive attribute group in the LoA-AHS hierarchy. These steps are done offline prior to the running of the authentication service.

At this point, the weightings are determined for the entire additive attributes groups in the authentication environment. However in an authentication instance, there are cases in which only some of the additive attributes are used. For example, a user uses 'ID Credential-based Authentication' and 'Location-based Authentication' in an authentication instance. In this case, the weightings $W' = [0.644, 0.085, 0.271]$ for Set A = {'ID Credential-based Authentication', 'Location-based Authentication', 'Assertion-based Authentication'} need to be modified to reflect the weightings for Set B = {'ID Credential-based Authentication', 'Location-based Authentication'}. Again, this is done by taking the weightings [0.644, 0.085] for 'ID Credential-based Authentication' and 'Location-based Authentication' in Set A and normalising it to weightings $W'' = [0.883, 0.117]$ for Set B. This process is dynamic and does not involve the judgement of the decision-maker therefore it is a real-time process to

suit different authentication instances.

Once the weightings of the attributes are determined, they should be integrated into the derivation algorithm. We do this by integrating the weighting of an attribute into its component LoA. The method is described in the following.

Assume that w_i is attribute i's weighting, LoA_i is its original component LoA value, and LoA_{ai} is i's adjusted LoA value. The adjusted LoA value is the one that has taken into account the effect of w_i on LoA_i. In other words, the effect of attribute i's weighting on the final aggregate LoA is embedded into its adjusted component LoA value, LoA_{ai}. So $W = \{w_1, w_2, ..., w_n\}$, $LoA = \{LoA_1, LoA_2, ..., LoA_n\}$, and $LoA_a = \{LoA_{a1}, LoA_{a2}, ..., LoA_{an}\}$ now denote the set of weightings, original component LoA values, and weighting adjusted component LoA values for n additive attributes (n>1), respectively.

It is worth noting that the sum of the weightings of all of the attributes in an additive group is always 1. Thus, without specifying a weighting for each of the attributes, an additive algorithm initially assumes that each of the *n* attributes in a group has the same level of contribution (weighting), i.e. each attribute initially has a weighting equal to 1/n. Then the adjusted weighting for attribute i will be the difference between the real weighting, w_i, and the assumed weighting, 1/n. w_i may be greater than, less than or equal to 1/n. If $w_i = 1/n$, attribute i is said to contribute the same as it initially does, therefore its adjusted component LoA value is the same as its original component LoA value. If $w_i > 1/n$, attribute i contributes more to the aggregate LoA than it initially does. Reflecting its component LoA value, the adjusted component LoA is its original component LoA plus the difference, and vice versa if $w_i < 1/n$. Therefore, the adjusted component LoA value for attribute i can be estimated using the following equation:

$$LoA_{ai} = LoA_i * (1 + (w_i - \frac{1}{n})) \qquad (4.2)$$

By integrating attributes' weightings into their respective component LoA values, the adjusted component LoA values can capture the effects of the attributes on the overall authentication assurance level of an authentication event in a more accurate manner. By taking the adjusted component LoA values of the attributes involved in an authentication instance as input, a more precise Agg-LoA value could be derived using an aggregate LoA derivation algorithm. The next two subsections describe two such algorithms. The first algorithm is built on Subjective Logic, and the second one is built on Probability Theory, thus these algorithms are, called the Subjective Logic-based aggregate LoA derivation algorithm and the Probability Theory-based aggregate LoA derivation algorithm respectively.

4.4.2 Algorithm One: Subjective Logic-based Aggregate LoA Derivation Algorithm ($ALoA_{AD}$-SL)

4.4.2.1 Subjective Logic for Aggregate LoA Derivation

In standard logic, propositions that are considered either true or false are binary. However, Josang believes that the absolute certainty of a statement cannot be determined in the real world. Since the statements are always assessed by individuals, they can never be considered as representing complete general and objective opinions [Josa00]. Therefore, Joang has defined a framework called 'subjective logic' that consists of a belief model, named an opinion model, and a set of operations for combining opinions. Subjective logic is developed to mathematically describe and manipulate subjective beliefs; it is an extension of standard logic that uses continuous uncertainty and belief parameters instead of only using discrete truth values.

The Opinion Model

The opinion model defines an opinion about a proposition, which translates into degrees of belief or disbelief. In addition to belief and disbelief, it is necessary to

take into consideration degrees of ignorance, which can be interpreted as the lack of evidence to support either belief or disbelief [Joan00].

For a single opinion about a proposition, the model has:

$$b + d + i = 1, \{b, d, i\} \in [0,1] \qquad (4.3)$$

where b, d and i designate belief, disbelief and ignorance, respectively.

Figure 4.2 shows the opinion model triangle, where an opinion can be uniquely described as a point {b, d, i} on the triangle. For example, the bottom line between belief and disbelief represents situations with zero ignorance and is equivalent to a traditional probability model, i.e. the opinion {0.5, 0.5, 0} means there are equally strong reasons to believe that the proposition is true as false; whereas the opinion {0.4, 0.5, 0.1} indicates there are the reasons 40% to believe, 50% to disbelieve and 10% uncertain about the proposition.

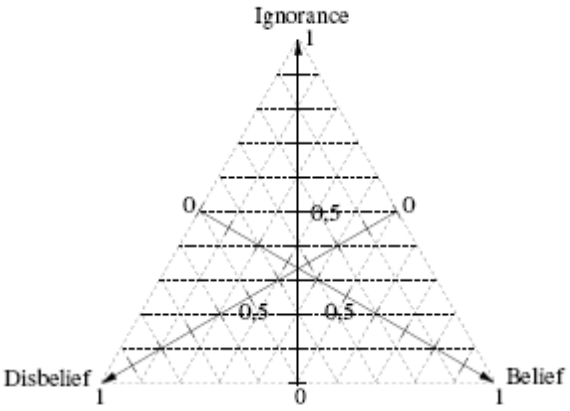

Figure 4.2 Opinion Triangle

In this opinion model, various operations for processing multiple opinions are defined, including conjunction, disjunction, negation, consensus, recommendation and ordering. The following introduces the consensus operation, given that it is the most relevant operation for the derivation of an aggregate LoA of additive LoA-effecting attributes.

Consensus between Independent Opinions

The consensus rule for combining independent opinions consists of combining two or more independent opinions about the same proposition into a single opinion. In [Joan00], the following definition is proposed:

Definition 4 – Let $\pi_p^A = \{b_p^A, d_p^A, i_p^A\}$ and $\pi_p^B = \{b_p^B, d_p^B, i_p^B\}$ be opinions held by agents A and B, respectively, about the same proposition p. Let $\pi_p^{A,B} = \{b_p^{A,B}, d_p^{A,B}, i_p^{A,B}\}$ be the opinion such that

$$b_p^{A,B} = (b_p^A i_p^B + b_p^B i_p^A) / k$$

$$d_p^{A,B} = (d_p^A i_p^B + d_p^B i_p^A) / k$$

$$i_p^{A,B} = (i_p^A i_p^B) / k,$$

where $k = i_p^A + i_p^B - i_p^A i_p^B$ such that $k \neq 0$.

Then $\pi_p^{A,B}$ is called the Bayesian consensus between π_p^A and π_p^B, representing an imaginary agent [A, B]'s opinion about p. This operation uses operator $\oplus$. The consensus operation can be expressed as $\pi_p^{A,B} = \pi_p^A \oplus \pi_p^B$. Note that two opinions both containing zero ignorance cannot be calculated by a consensus operation. This is explained by interpreting ignorance as room for influence, meaning that it is only possible to influence an opinion that has not yet been committed to belief or disbelief.

From the above discussion, it can be seen that subjective logic holds that it is impossible to determine with absolute certainty whether some propositions are true or false. This concept matches the case of authentication LoA, i.e. the authentication will not always produce a perfect outcome. By defining an opinion model and defining a set of operations (e.g. consensus operation) to process multiple opinions,

subjective logic can be used to make a binary proposition into a quantifiable statement and provides a way to derive a consensus opinion of two independent opinions.

One of the biggest challenges in the design of an aggregate LoA derivation algorithm is determining how to combine the assurance levels of multiple additive (i.e. independent) LoA-effecting attributes to form a unified aggregate LoA. By applying the opinion model to the algorithm design, the additive LoA-effecting attributes can effectively be transformed as 'opinions' about the same proposition (e.g., "The user's identity as A can be verified,"). A single aggregated opinion (i.e. an aggregate LoA) can then be computed by applying the consensus operation. More details on how to transform the component LoA values of multiple LoA-effecting attributes into subjective logic 'opinions' and how to use the consensus operation to calculate an aggregate LoA value based upon the opinions are presented in the next section.

4.4.2.2 ALoA$_{AD}$-SL Algorithm Design

Using the SLO model, each of the additive attributes is transformed into an 'opinion' in the opinion model. For example, an attribute x's opinion about the aggregated authentication assurance level p can be expressed as:

$$\pi_p^x = b + d + u = 1, \{b, d, u\} \in [0, 1] \qquad (4.4)$$

Where π is the opinion function, p is the proposition which π has an opinion of (in this case, p refers to the aggregate LoA), x is the attribute, and b, d, and u represent belief, disbelief and uncertainty, respectively.

We now need to determine the values for the tuple < b, d, u>. Belief b refers to the level of trust in attribute x's opinion. It is set to a value in the range [0,1], where 0 stands for no trust at all and 1 stands for absolute certainty. The level of trust in an authentication outcome (i.e. the meaning of b) obviously has a similar meaning to

the component LoA (which refers to the level of confidence in an authentication outcome). However, as LoA values are scoped between 1 and 4, and b in the subjective logic uses a scale from 0 to 1, we need a transformation method to transform LoA values from the scale of [1, 4] to values in the scale of [0, 1]. This scale transformation is done using the following mapping: $b(0.25) = LoA_x(1)$, $b(0.5) = LoA_x(2)$, $b(0.75) = LoA_x(3)$, and $b(1) = LoA_x(4)$.

Disbelief d refers to the level of inaccuracy in attribute x's opinion. It is usually used to measure the inaccuracy (hardware fault) of some hardware-based attributes, such as the case in sensor network and hardware location-based authentication. Therefore, an inaccuracy level d needs to be specified for such authentication attributes. On the other hand, unlike hardware-based authentication attributes, credential-based authentication attributes only have belief and uncertainty values, but not accuracy value. This is because, for credential-based authentication, if the authentication outcome is successful, then the level of accuracy is taken as 100% (i.e. d = 0).

Based upon these considerations, we can define the opinion for attribute x as follows: $\pi_p^x = \begin{cases} b = LoA_x \\ d = m \\ u = 1 - LoA_x - m \end{cases}$, in which, m is the inaccuracy parameter, and b+d+u = 1.

Once the opinions of all the attributes involved are defined, we can calculate a combined opinion by using the consensus operation defined in Definition 4. For an additive attributes set with more than two attributes, a recursion operation is used to calculate the final aggregate LoA. The pseudo code for this algorithm is given below.

```
INPUT: Additive LoA-effecting attributes Set N, N >= 1

OUTPUT: Agg-LoA Value Agg-LoA
```

```
Integer Method Additive_LoA(Object Set N)
      READ attribute LoA values in N;
         SAVE into Integer ARRAY LoA[];
      READ attribute weightings in N;
         SAVE into Integer ARRAY weighting[];
   SET LoA_Length to the Length of ARRAY LoA[];
   SET Agg-LoA to the aggregate LoA value;
   /* if Set N contains only one attribute, the component LoA
value of this attribute is returned */
   IF LoA_Length IS equals to 1 THEN
      RETURN LoA[0];
   /* if Set N contains more than one attribute, calculate
adjusted component LoA value based on the weightings*/
   ELSE IF LoA_length is more than 1 THEN
      REPEAT
         LoA[x] = LoA[x]*(1 + weighting[x]-(1/LoA_length));
         SET X to X+1;
      UNTIL x equals to LoA_length;
    /* calculate the agg-LoA based on first two attributes */
      CALL Agg-LoA = Add_LoA_AD(LoA[0], LoA[1]);
      SET index to 2;
      IF index equals to LoA_length THEN
         RETURN Agg_LoA;
/* check if LoA_length is more than 2, recursion operation */
      ELSE
       REPEAT
         CALL Agg-LoA = Add_LoA_AD(Agg_LoA, LoA[index]);
         SET index to index+1;
       UNTIL index equals to LoA_length
       RETURN Agg_LoA;
```

```
      ENDIF
   ENDIF
/* method to calculate Agg-LoA */
Integer Method Add_LoA_AD(Object LoA_a, Object LoA_b)
      SET belief_a to LoA_a;
      SET uncertain_a to 1- belief_0;
      SET belief_b to LoA_b;
      SET uncertain_b to 1- belief_1;
      DEFINE
      π_p^a = {belief_a, 0, uncertain_a}
                  π_p^b = {belief_b, 0, uncertain_b}
                  Compute
                  k = uncertain_a + uncertain_b − uncertain_a * uncertain_b
                  belief_{a,b} = (belief_a * uncertain_b + belief_b uncertain_a) / k
      RETURN belief_{a,b};
```

Table 4.3 The pseudo code of the $ALoA_{AD}$-SL algorithm

An implementation of this algorithm is in Appendix A and the evaluation of this algorithm is undertaken in Section 4.5.

4.4.3 Algorithm Two: Probability Theory-Based Aggregate LoA Derivation Algorithm ($ALoA_{AD}$-PT)

The derivation of an aggregate LoA value given the component LoA values of a set of additive LoA-effecting attributes may also be done based on a Probability Theory (PT). The design of such a PT-based aggregate LoA derivation algorithm, i.e. $ALoA_{AD}$-PT algorithm is more straightforward when compared to the $ALoA_{AD}$-SL algorithm. This method employs the inclusion-exclusion principle of probability theory to calculate the consensus of multiple additive LoA-effecting attributes. For a

set of additive LoA-effecting attributes $S = \{a_1, a_2, ...a_n\}$, the probability of an attribute $a_i \subseteq S$ is represented as a number $P(a_i) \subseteq [0,1]$. In other words, $P(a_i)$ is the probability of the trustworthiness of an attribute a_i ,which declares the authenticity of someone's identity. It is obvious that the meaning of $P(a_i)$ and the meaning of the component LoA value LoA_{ai} are similar. Since component LoA values are scoped between 1 and 4, and $P(a_i)$ in the probability theory uses a scale from 0 to 1, similar to the mapping mechanism we used in the $ALoA_{AD}$-SL design, we transform LoA values from the scale of [1, 4] to values in the scale of [0, 1] by using the following mapping, P(0.25) = $LoA_x(1)$, P(0.5) = $LoA_x(2)$, P(0.75) = $LoA_x(3)$, and P(1) = $LoA_x(4)$.

For independent events (attributes in our case) $\{a_1, a_2, ...a_n\}$ in a set S, the inclusion-exclusion principle is given in equation (4.5) below when n=2,

$$P(a_1 \cup a_2) = P(a_1) + P(a_2) - P(a_1)P(a_2) \qquad (4.5)$$

In equation (4.5), $P(a_1 \cup a_2)$ denotes the consensus of trustworthiness of attribute a_1 and a_2 that declares the authenticity of someone's identity. When n > 2, it can be extended to the general equation of the inclusion-exclusion principle, as expressed in (4.6).

$$P(\bigcup_{i=1}^{n} a_i) = \sum_{k=1}^{n} P(a_i) - \sum_{i,j:i<j} P(a_i \cap a_j) + \sum_{i,j,k:i<j<k} P(a_i \cap a_j \cap a_k) - ... + (-1)^{n-1} P(\bigcap_{i=1}^{n} a_i) \qquad (4.6)$$

In order to simplify the equation (4.6), we use the complement rule, i.e. for the entire event A in a set S, $P(\overline{A})$ is the complement of $P(A)$, where

$$P(\overline{A}) = 1 - P(A) \qquad (4.7)$$

When n=2, we combine equations (4.5) and (4.7), i.e.

$$P(\overline{a_1 \cup a_2}) = 1 - P(a_1 \cup a_2) = 1 - P(a_1) - P(a_2) + P(a_1)P(a_2) = (1 - P(a_1))(1 - P(a_2))$$

Then, we have

$$P(a_1 \cup a_2) = 1 - P(\overline{a_1 \cup a_2}) = 1 - (1 - P(a_1))(1 - P(a_2)) \quad (4.8)$$

Similarly, when n=3, we have

$$P(a_1 \cup a_2 \cup a_3) = 1 - (1 - P(a_1))(1 - P(a_2))(1 - P(a_3)) \quad (4.9)$$

Thus,

$$P(\bigcup_{i=1}^{n} a_i) = 1 - (1 - P(a_1))(1 - P(a_2))...(1 - P(a_n)) = 1 - \prod_{i=1}^{n}(1 - P(a_i)) \quad (4.10)$$

Therefore, the $ALoA_{AD}$ algorithm is

$$ALoA_{(AD,level\_K)} = 1 - \prod_{i=1}^{n}(1 - P(a_i)) \quad (4.11)$$

To illustrate the use of this algorithm, suppose there are two additive LoA-effecting attributes at a particular level of LoA-AHS: $S = \{a_{username/password} = 2, a_{location} = 2\}$ in the scale of [1, 4]. We then use the mapping mechanism described above to compute the corresponding mapped value $\{P(a_{username/password}) = 0.5, P(a_{location}) = 0.5\}$ in the scale of [0, 1]. By applying equation (4.11), the calculated aggregate LoA value for this set of additive attributes is 0.75, which is assurance Level 3. In other words, the aggregated authentication assurance achieves level 3 by combining two Level 2 authentication methods. The pseudo code of this algorithm is given in Table 4.4.

```
INPUT: Additive LoA-effecting attributes Set N, N >= 1
OUTPUT: Agg-LoA Value Agg-LoA

Integer Method Additive_LoA(Object Set N)
      READ attribute LoA values in N;
         SAVE into Integer ARRAY LoA[];
      READ attribute weightings in N;
```

```
         SAVE into Integer ARRAY weighting[];
   SET LoA_Length to the Length of ARRAY LoA[];
   SET Agg-LoA to the aggregate LoA value;
   /* if Set N contains only one attribute, the component LoA
value of this attribute is returned */
   IF LoA_Length IS equals to 1 THEN
      RETURN LoA[0];
   /* if Set N contains more than one attribute, calculate
adjusted component LoA value based on the weightings*/
   ELSE IF LoA_length is more than 1 THEN
      REPEAT
         LoA[x] = LoA[x]*(1 + weighting[x]-(1/LoA_length));
         SET X to X+1;
      UNTIL x is equal to LoA_length;
    /* calculate the agg-LoA */
      SET variable index to 0;
      SET variable temp to 1;
      REPEAT
         temp=temp*(1-LoA[index]);
         SET index to index+1;
      UNTIL index equals to LoA_length
      SET Agg_LoA = temp;
      RETURN Agg_LoA;
```

Table 4.4 The pseudo code of the $ALoA_{AD}$-PT algorithm

An implementation of this algorithm is in Appendix A, and the evaluation of this algorithm is undertaken in Section 4.5.

4.5 Algorithms Evaluation

This section evaluates the three aggregate LoA derivation algorithms with regard to performance and the satisfactory rates of the derived aggregate LoA values. The performance is measured in terms of their computational cost. The satisfactory rate

is measured by comparing the derived aggregate LoA values to the algorithms with the anticipated values. Two sets of experiments are conducted for the evaluation. The first set assesses the computational time consumed by each of the algorithms, and the second set calculates the aggregate LoA and compare the values on different sets of LoA-effecting attributes with different component LoA values.

4.5.1 Evaluation Environment

The experiments are hosted on a Windows 7 OS running on a ThinkPad laptop with two 2.53GHz processors and 3072MB of memory. The prototype is implemented as a JAVA application with Java SE™ Runtime Environment version 6. The performance evaluation tool used is NetBeans IDE v6.8, which provides a wide range of profiling instruments for measuring CPU and memory costs.

4.5.2 Algorithms Performance Evaluation

The performance evaluation is done by measuring computational times consumed by the algorithms. To ensure statistical significance, the number of iterations (denoted as *n*) over which the performance is measured need to be determined. Three experiments have been conducted; each sets *n* to a different value and the computational times are measured in milliseconds. Figures 4.3 to 4.5 show the results of the experiments for the three algorithms: $ALoA_{WL}$, $ALoA_{AD}$-SL and $ALoA_{AD}$-PT, respectively. From the figures we can see that the bigger the value of *n*, the less of an effect arbitrariness has on the algorithm execution time. The linear regression line in the figures suggests an *n* value higher than 2.2k (i.e. 2200 iterations) is sufficient for $ALoA_{WL}$ algorithm, an *n* value higher than 2.3k (i.e. 2300 iterations) is sufficient for $ALoA_{AD}$-SL algorithm and an *n* value higher than 2.5k (i.e. 2500 iterations) is sufficient for $ALoA_{AD}$-PT algorithm. Thus an *n* value of 3000 is used in all the experiments.

Prior to performing the experiments, two main results were anticipated. Firstly, for

all three algorithms, the larger the input attribute set size is, the longer it would take to calculate the Agg-LoA value. This is because none of the algorithm's computational complexity is $O(1)$ (i.e. constant time and not related to the length of the input), which means the computational time grows when input size grows. Secondly, the computational times of the $ALoA_{AD}$-SL and $ALoA_{AD}$-PT algorithms would grow at the same rate and the computational time of $ALoA_{WL}$ would grows at a faster rate in comparison to the other two. This is because the time complexity of the $ALoA_{WL}$ algorithm is $O(n*\log(n))$ [Javaapi], which is 'linearithmic time', and the time complexity of both the $ALoA_{AD}$-SL and $ALoA_{AD}$-PT algorithms is $O(n)$, which is 'linear time'. For the same increment of input size n, 'linearithmic time' grows faster than 'linear time'.

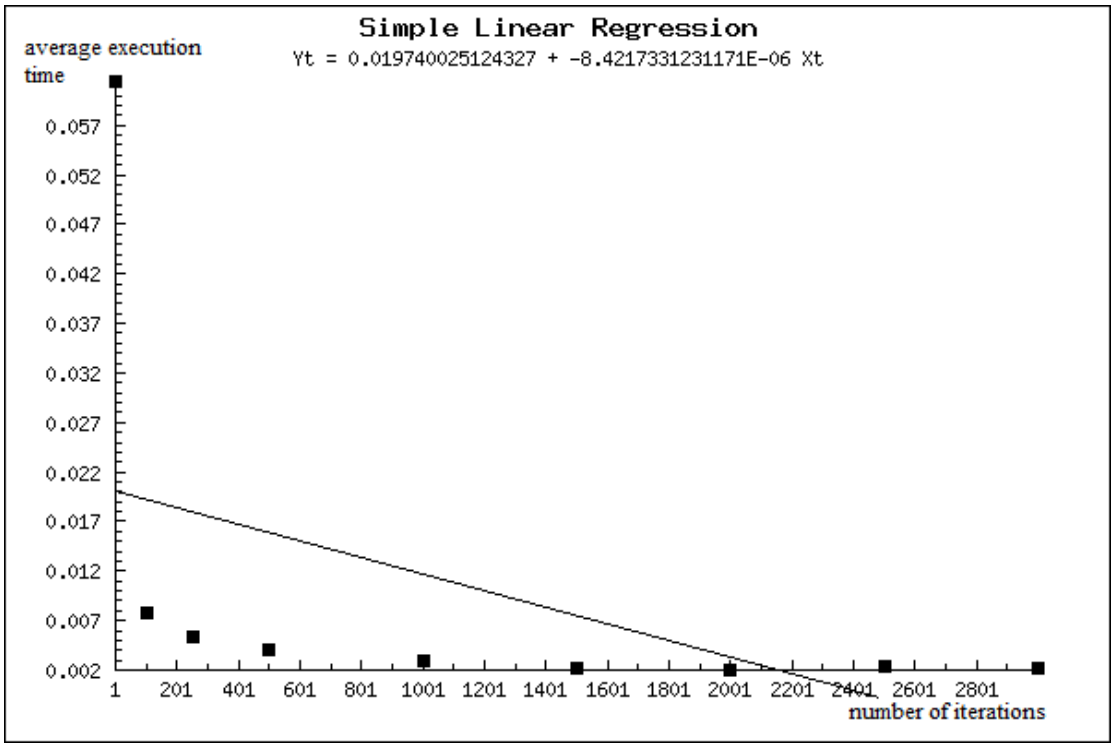

Figure 4.3 Linear Regression to determine number of iterations in $ALoA_{WL}$ Algorithm

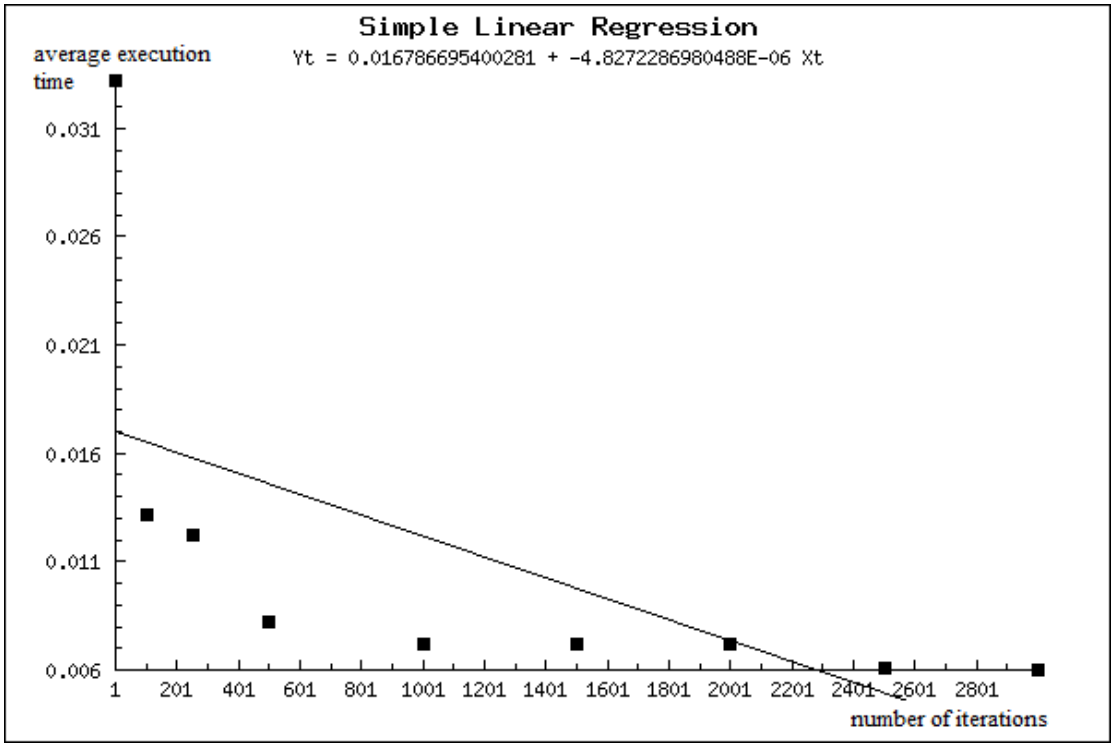

Figure 4.4 Linear Regression to determine number of iterations in $ALoA_{AD}$-SL Algorithm

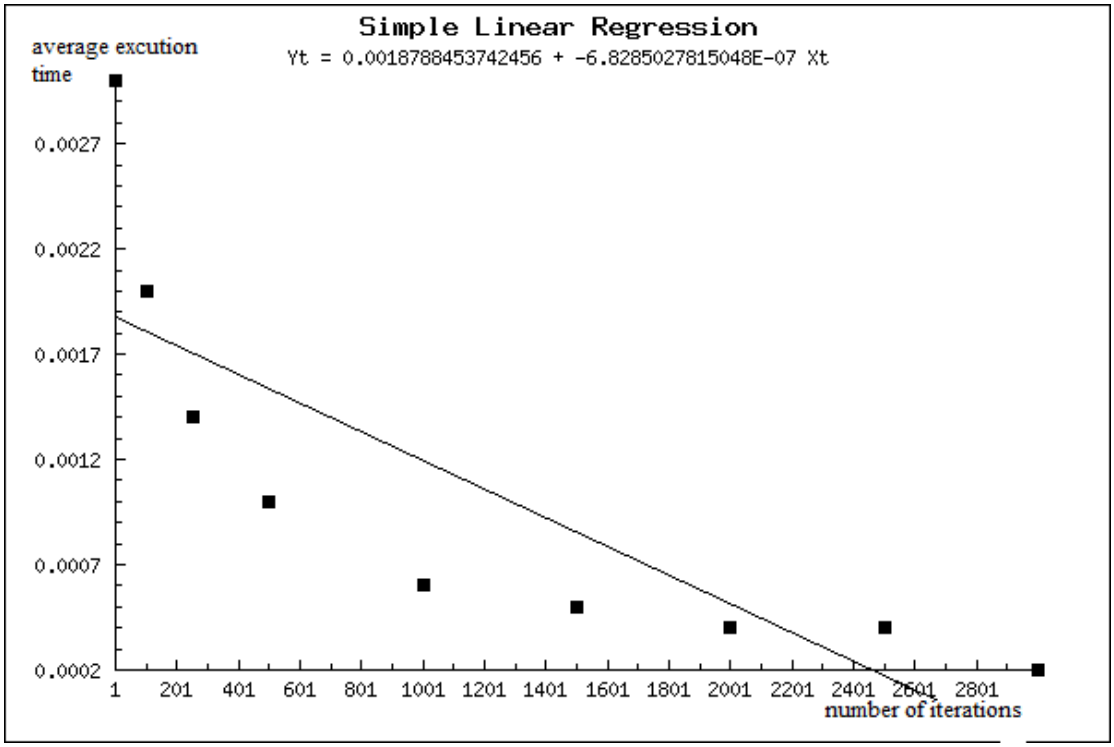

Figure 4.5 Linear Regression to determine number of iterations in $ALoA_{AD}$-PT Algorithm

The algorithm performance evaluation experiments took six sets of LoA-effecting attributes as input and measured the computational cost of each set on each of the algorithms. Figure 4.6 shows the results of the experiments. It shows three sets of

execution times in milliseconds against five inputs. The three sets are from the weakest link $AloA_{WL}$, additive $ALoA_{AD}$-PT and $ALoA_{AD}$-SL, respectively. Five inputs contain 1, 2, 5, 10 and 20 LoA-effecting attributes respectively. Three observations can be made from these results. First, the execution times increase in the three algorithms as the input size increases. This observation is in line with our expectations. The second observation is that when input size grows, the results for the $ALoA_{WL}$ algorithm increase faster than the $ALoA_{AD}$-SL and $ALoA_{AD}$-PT algorithms. This result is also within our anticipations. The third observation is that the results for the $ALoA_{AD}$-SL algorithm are slightly bigger than that for the $ALoA_{AD}$-PT algorithm. This is due to the fact that the calculation of the $ALoA_{AD}$-SL algorithm is slightly more complex than the $ALoA_{AD}$-PT algorithm, as shown in Tables 4.2 and 4.3. Therefore, though the computational complexity of both algorithms is in the same magnitude, the execution time of the $ALoA_{AD}$-SL algorithm is longer than that of the $ALoA_{AD}$-PT algorithm.

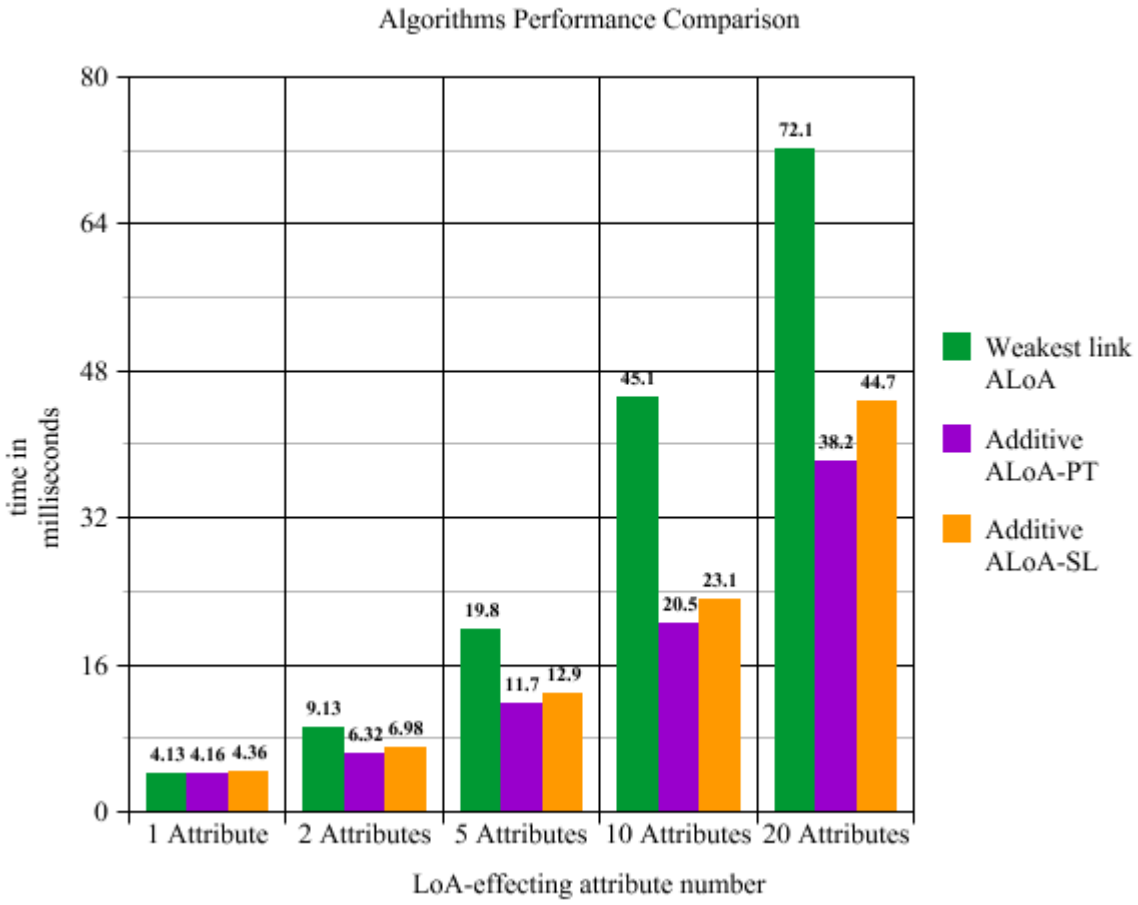

Figure 4.6 Algorithms performance comparison

4.5.3 Algorithm Satisfactory Rate Evaluations

As shown in Table 4.5, evaluations are done given five sets of LoA-effecting attributes as input values. The anticipated results are listed next to the input values. The calculated results using $ALoA_{WL}$, $ALoA_{AD}$-PT and $ALoA_{AD}$-SL algorithms are listed in columns 3, 4 and 5 respectively.

From the table, it can be seen that the algorithms returned satisfactory results. In input set A, only one attribute is considered, thus the anticipated result is the attribute's component LoA value. In input sets B, C and D, $ALoA_{WL}$ returned the aggregate LoA values, which are the lowest values among all the attributes' component LoA values. This result is satisfactory. The $ALoA_{AD}$-PT and $ALoA_{AD}$-SL algorithms returned aggregate LoA values that are larger than the largest individual component LoA values from the input sets, which is also satisfactory. In

input set E, both $ALoA_{AD}$-PT and $ALoA_{AD}$-SL algorithms give a result of 1. This is because set E contains an attribute with a component value of 1, which means absolute certainty in the $ALoA_{AD}$-SL algorithm and 100% trustworthiness in the $ALoA_{AD}$-PT algorithm. Therefore, no matter what the other attributes' component LoA values are, as long as they are in an additive relationship, the largest aggregate LoA value is 1. To sum up, the experiments showed a 100% satisfactory rate of the derived aggregate LoA values for all of the algorithms under various input values.

Input	**Anticipated results**	**$ALoA_{WL}$**	**$ALoA_{AD}$-PT**	**$ALoA_{AD}$-SL**	**Agg-SR**
A {a_1=0.75}	$ALoA_{WL}$=0.75 $ALoA_{AD}$-SL=0.75 $ALoA_{AD}$-PT=0.75	0.75	0.75	0.75	100%
B {b_1=0.25, b_2= 0.5}	$ALoA_{WL}$=0.25 $ALoA_{AD}$-SL>0.5 $ALoA_{AD}$-PT>0.5	0.25	0.625	0.571	100%
C {c_1=0.25, c_2=0.5, c_3=0.75}	$ALoA_{WL}$=0.25 $ALoA_{AD}$-SL>0.75 $ALoA_{AD}$-PT>0.75	0.25	0.9	0.8125	100%
D{ d_1=0.25, d_2=0.5, d_3=0.75, d_4=0.25}	$ALoA_{WL}$=0.25 $ALoA_{AD}$-SL>0.75 $ALoA_{AD}$-PT>0.75	0.25	0.92	0.83	100%
E { d_1=0.25, d_2=0.5, d_3=1}	$ALoA_{WL}$=0.25 $ALoA_{AD}$-SL=1 $ALoA_{AD}$-PT=1	0.25	1	1	100%

Table 4.5 Satisfactory rate of LoA derivation algorithms results

4.6 Chapter Summary

This chapter has presented the design of three aggregate LoA derivation algorithms: the weakest link $ALoA_{WL}$ algorithm, the subjective logic-based additive $ALoA_{AD}$-SL algorithm, and the probability theory-based additive $ALoA_{AD}$-PT algorithm. For the aggregate LoA derivation when attributes are in an additive relationship, we have also presented a method by which different weightings of different attributes can also be taken into account. The performance and satisfactory rate of these algorithms have been evaluated via experiments. In the next chapter, we will present the design of the GEA-LoADM Model; the algorithms described in this chapter are part of this model.

Chapter 5

A Generic E-Authentication LoA Derivation Model

In the previous chapter, we described three aggregate LoA derivation algorithms. This chapter integrates these algorithms to devise our approach to real-time aggregate LoA derivation, the Generic E-authentication LoA Derivation Model (GEA-LoADM).

5.1 Design Requirement Specifications

The design of the GEA-LoADM model is non-trivial, and the following issues should be resolved: 1) identify all LoA-effecting attributes and determine their component LoA values in an authentication environment; 2) construct an LoA-AHS structure based on the identified LoA-effecting attributes and allocate weightings to each of the attributes in the additive relationship; 3) reliably acquire dynamic real-time LoA-effecting attributes (i.e. the contributing LoA-effecting attributes) in an authentication instance; and 4) derive an aggregate LoA value based upon the contributing LoA-effecting attributes in a systematic manner. The model should be flexible enough to accommodate any changes in the attribute set, and the resulting weighting changes for the set. We have identified in detail the following design requirements for the GEA-LoADM:

R1 - Extensibility:

The model supports the use of an extensible set of LoA-effecting attributes. The current set of LoA-effecting attributes (Table 3.2) is decomposed from a range of existing works [Burr06, Samlac] and Grid authentication scenarios/models. In the future, there may be new or emerging authentication technologies that could bring in new LoA-effecting attributes, so, the design of the model should allow new

attributes to be easily added in and obsolete ones to be removed from it.

R2 – Support the use of policy-driven attribute LoA values:

Each of the LoA-effecting attributes will have a component LoA associated with it. These component LoA values are usually dependent on the LoA regime used, which is expected to be defined by the relevant international communities. To support the use of any, either existing (such as the NIST definition [Burr06]) or emerging, LoA regimes, the model will use a policy-driven component to process component LoA values.

R3 – Systematic determination of weightings for LoA-effecting attributes:

An authentication process may involve a number of LoA-effecting attributes. Some of these attributes may be intangible or their impacts difficult to measure and quantify. For such attributes, the GEA-LoADM model should use a scientific approach to determine their impacts (i.e. weightings) on the aggregated assurance level of an entire authentication instance.

R4 – Derivation of aggregate LoA:

Given a set of LoA-effecting attributes, their types of relationships and their component LoA values and weightings, the model should be able to derive an aggregate LoA for an authentication instance.

R5 – LoA conveyance:

The model should be able to convey to a relying party the aggregate LoA, the LoA-effecting attributes' names and any other items that the CSP has verified. The relying party can then use this information to make access control or authorisation decisions.

R6 – Modular structure:

The model should employ a modular structure such that any change made to one architectural module does not lead to any changes in other modules. Each module

(component) of the model is designed to fulfil a specific function and can be easily plugged in or taken out.

R7 – Security:

The application of the model should not weaken or lower the level of security afforded by the original access control service (i.e. before applying the GEA-LoADM model).

R8 – Performance:

Adequate considerations should be given to the design of the model to reduce the level of overhead and run-time delay as introduced by the LoA derivation service provided by the GEA-LoADM model.

5.2 Architecture Overview

Figure 5.1 shows the GEA-LoADM architecture. From the figure, it can be seen that it has a number of architectural components. These components can largely be classified into the following groups: an off-line component, a real-time component and an LoA-effecting attributes policy database. The output of the GEA-LoADM is an Agg-LoA value calculated for an authentication instance performed by a requester. This value is consumed by a relying party (i.e. a service provider), which can be either a shibboleth attribute authority [Shib05] or an authorisation decision engine to grant access to the requester.

The off-line component is an LoA-effecting Attributes Policy Manager (LoA-APM). It is responsible for identifying a set of LoA-effecting attributes in an authentication environment, structuring them into a hierarchy based on their mutual relationships, and calculating the weightings for the additive attributes in the set. The component comprises two further functional modules: an LoA-effecting Attributes Hierarchical Structure (LoA-AHS) and an LoA-effecting Attributes Weightings Allocation Module (LoA-AWAM). The LoA-AHS is responsible for identifying a set of LoA-

effecting attributes in a given authentication environment, constructing a hierarchical LoA-effecting attributes structure (such as the one shown in Figure 5.3), and categorising the attributes into different groups and levels based on their mutual relationships. These tasks are expected to be undertaken manually by an authentication administrator or access policy decision-maker, based on their security policies and access control requirements. The LoA-AWAM is responsible for calculating the LoA weightings for additive LoA attributes. The weightings, the attribute hierarchical structure, the indicators of the relationships among different attributes, and the component LoA values are all then stored in an LoA-effecting attributes policy database (LoA-APDB). The working mechanisms of, and the methodology used in the design of, these functional modules are detailed in Section 5.3.1.

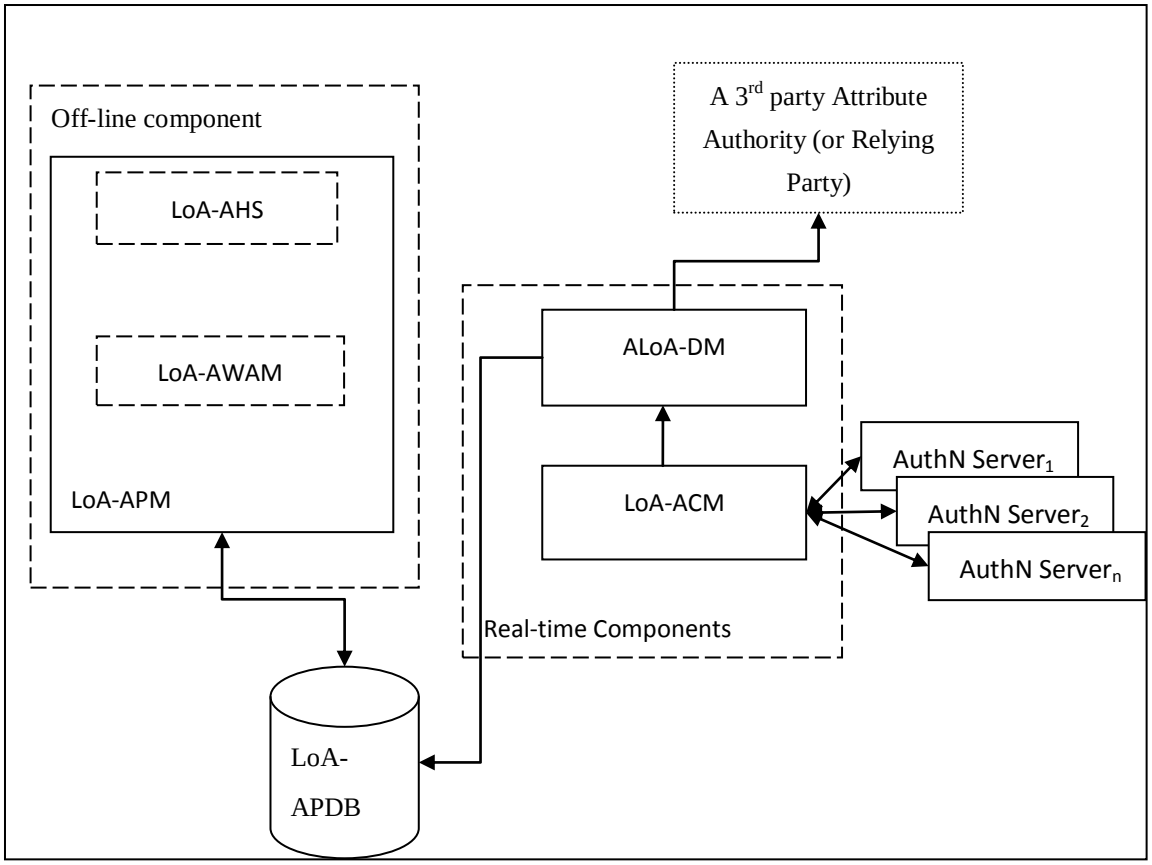

Figure 5.1. GEA-LoADM architecture

The real-time component has two functional modules: an LoA-effecting Attributes Collection Module (LoA-ACM) and an Authentication LoA Derivation Module

(ALoA-DM). The LoA-ACM performs three tasks. It first receives a set of contributing LoA-effecting attributes involved in an authentication instance from the authentication services in a SAML assertion format. It then parses the assertions and retrieves the contributing LoA-effecting attributes' names from it. Next, the LoA-ACM sends the retrieved attributes' names to the ALoA-DM. The ALoA-DM receives the contributing LoA-effecting attributes' names and fetches both their and their parent attributes' names (the database stores the hierarchical structure), component LoA values, relationship indicators and weightings. This information is used to form a sub-LoA-AHS for this particular authentication instance. Then the ALoA-DM uses the sub-LoA-AHS as input to calculate the aggregate LoA using an authentication LoA derivation algorithm corresponding to the settings of the authentication environment. The design details of the LoA-ACM and ALoA-DM are described in Sections 5.3.3 and 5.3.4, respectively.

The LoA-APDB (LoA-effecting attributes policy database) is a database storing all of the LoA-effecting attributes identified by the LoA-AHS, their relationships, their component LoA values and the additive LoA attributes' weightings. The technical details of this module are described in Section 5.2.

As depicted by Figure 5.2, the working of the GEA-LoADM model is as follows:

Offline-Step 1: The authentication environment is analysed and all of the LoA-effecting attributes of the authentication environment are identified and constructed in the LoA-AHS structure. The corresponding component LoA values are estimated, too. Section 5.4.1.1 discusses this in detail.

Offline-Step 2: The LoA-AWAM module (as shown in Section 5.3.1.2) is used to estimate the weightings of the additive attributes.

At this point, all of the LoA-effecting attributes in the authentication environment have been structured, their component LoA values estimated and the weightings of the additive attributes calculated. All of the information is stored in the LoA-APDB.

The off-line steps are executed the first time the model is implemented in an authentication environment and should be reviewed or redone when there is a policy change (e.g. adjustment of the LoA values of LoA-effecting attributes; change of the authentication impact among additive attributes), or when new attributes are added or obsolete attributes omitted.

The following real-time steps are executed per authentication instance:

Real-time Step 1: Upon the execution of an authentication instance, the LoA-ACM receives the SAML assertions of the contributing LoA-effecting attributes from the authentication services for this instance.

Real-time Step 2: The LoA-ACM parses the assertions and retrieves the contributing LoA-effecting attributes' names and sends them to the ALoA-DM.

Real-time Step 3: The LoA-DM calls the LoA-APDB to fetch the attributes' component LoA values, the relationship indicators and the additive attributes weightings from the LoA-APDB. The attributes' parent attributes (based on the LoA-AHS) are also retrieved to form a sub-LoA-AHS for the authentication instance. The algorithm in the ALoA-DM (as discussed in Section 4.3 and 4.4) derives the aggregate LoA value based on the sub-LoA-AHS structure.

The decision-maker reviews the authentication events and the derived LoA values periodically. If there is a policy change or any abnormal situation such as faulty/unsatisfactory results, the decision-maker can revise the hierarchical structure and the weighting policies to make the appropriate modifications.

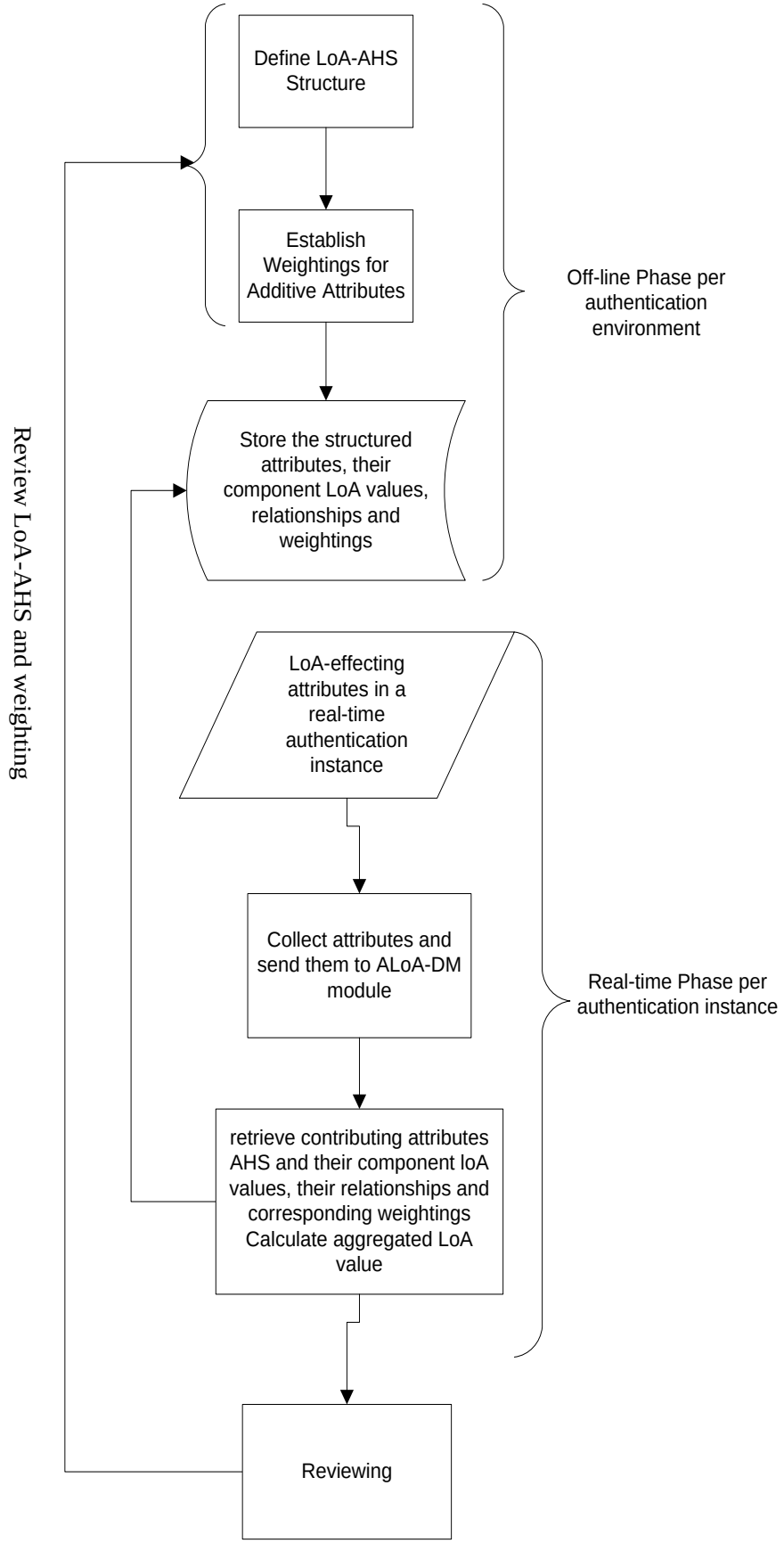

Figure 5.2 GEA-LoADM system execution flow

5.3 Architectural Components Design

5.3.1 LoA-effecting Attributes Policy Manager (LoA-APM)

The LoA-APM is proposed as an off-line component, meaning that its functions are performed prior to the execution of authentication procedures.

5.3.1.1 LoA-effecting Attributes Hierarchical Structure (LoA-AHS)

The LoA-AHS module is responsible for

- managing (i.e. adding, deleting and classifying) LoA-effecting attributes;
- assigning component (or attribute) LoA values to each of the attributes; and
- constructing the attributes into a hierarchical structure based on their mutual relationships.

The first two tasks are authentication environment-dependent. They are also dependent on access policies that are, in turn, influenced by factors such as the values of the asset under protection and the risks in the underlying access environment. We have examined and extended the attributes identified by NIST and OASIS and produced a generic set of LoA-effecting attributes, as described in Section 3.2. In addition, we have examined the mutual relationships among these attributes and organised them into the hierarchical structure shown in Figure 5.3. From the figure, it can be seen that the attributes at the same horizontal level enjoy one type of mutual relationship, either additive or weakest link. The aggregate LoA value for a group of related attributes (shown as circled with a dashed line) at that level is determined by the component LoA values of the attributes in that group, as well as their mutual relationship. This aggregate LoA, say level-k aggregate LoA, once calculated, is then fed into the level immediately above it, i.e. level (k+1), as the component LoA value of the attribute directly connected to the group. For example, as shown in the figure, the bottom group of two attributes, {‘Token Type’, ‘KeyStore’} located at level 5, are connected to the ‘Authentication Token’ attribute

at level 4. The component LoA value for the 'Authentication Token', called the $LoA_{AuthNToken}$, is then equal to the aggregate LoA value (named $LoA_{5,1}$, denoting the LoA value for the first group of attributes at level 5) of the two connected attributes at the level immediately below it. These two attributes have the weakest link mutual relationship; therefore, the aggregate LoA value of the group, $LoA_{5,1}$, is equal to *min*($LoA_{TokenType}$, $LoA_{KeyStore}$). $LoA_{5,1}$ is then taken as the component LoA value for the attribute, the 'Authentication Token', i.e. $LoA_{AuthNToken} = LoA_{5,1}$. Applying the same method to all of the attribute groups in the structure in a bottom-up manner, we will eventually get an LoA value for the root node; this root LoA value is the aggregate LoA value for the entire authentication instance.

This structured approach to LoA-effecting attributes' identification, classification, and organisation is an essential step towards the determination of their respective weightings on, and derivation of, the overall confidence level for an authentication instance, in a scientific manner. This structure has a number of additional merits. For example, it is flexible and extensible. Any emerging LoA-effecting attributes can easily be added into the structure, and any obsolete ones removed from it without affecting other levels in the hierarchy. Also, once constructed, an LoA-AHS for a given authentication environment will only need to be revised when there is a change in the authentication attributes at any level.

5.3.1.2 LoA-effecting Attributes Weighting Allocation Module (LoA-AWAM)

When calculating an aggregate LoA for a set of attributes that are in an additive relationship, their respective weightings should be determined first. The LoA-AWAM module uses the AHP's pair-wise comparison technique discussed in Section 4.4.1 to calculate the relative weightings of the attributes.

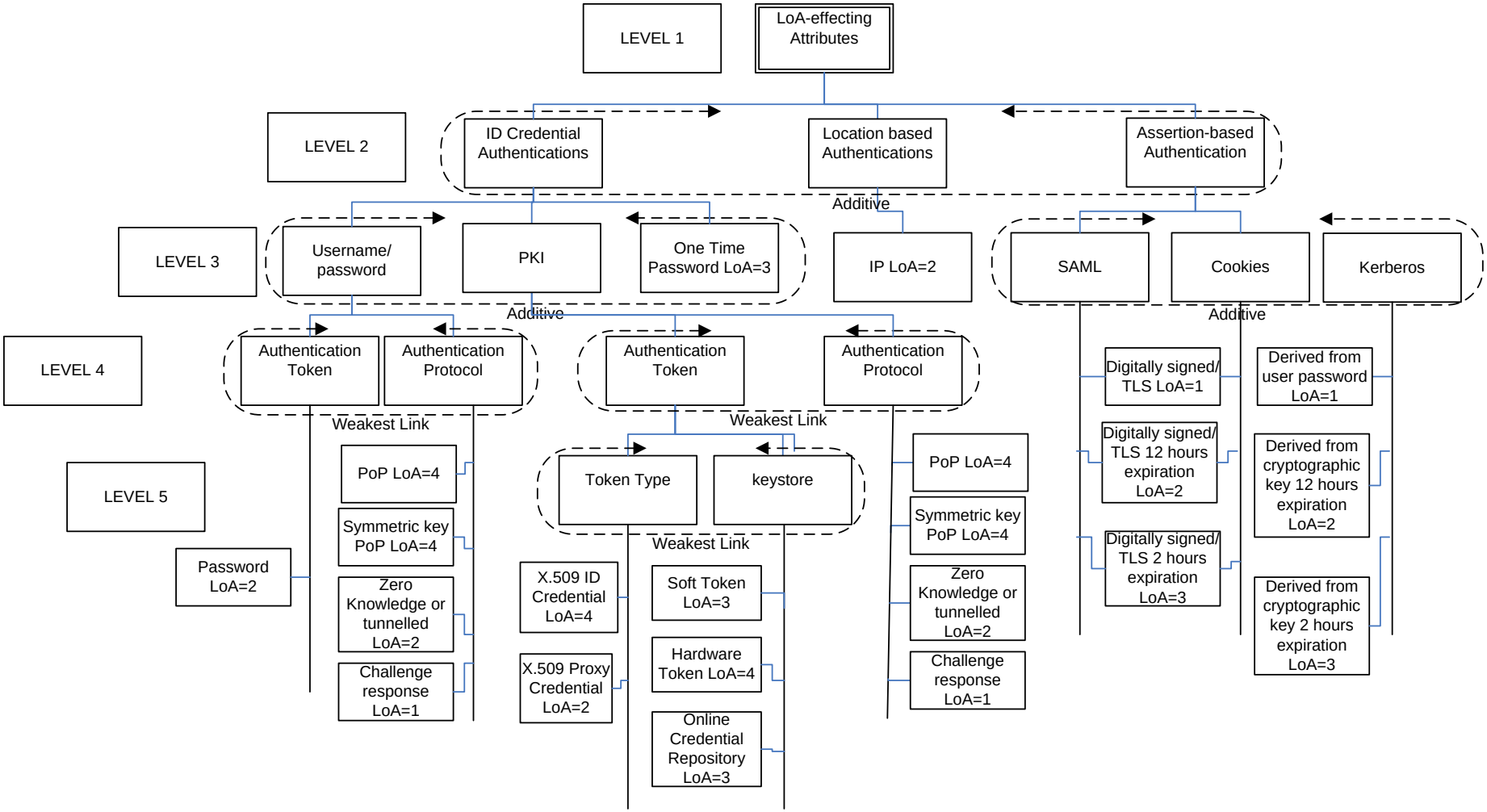

Figure 5.3 A generic LoA-AHS structure

5.3.2 LoA-effecting Attributes Policy Database (LoA-APDB)

The LoA-APDB is a database containing four tables. Table one, called the '*AHS*' table, stores the LoA-AHS structure, the 'Weightings' and the 'Relationship Types' of the LoA-effecting attributes. Table two, called the '*attributes*' table, stores 'Attribute Names' and 'Component LoA Values' of the LoA-effecting attributes. Table three, called the 'aggregate LoA' table, stores the aggregate LoA value derived by the ALoA-DM for each authentication instance. Table four, called the '*Contributing AHS*' table, links with the '*aggregate LoA*' table and stores the corresponding contributing LoA-effecting attributes in a hierarchal structure for each authentication instance.

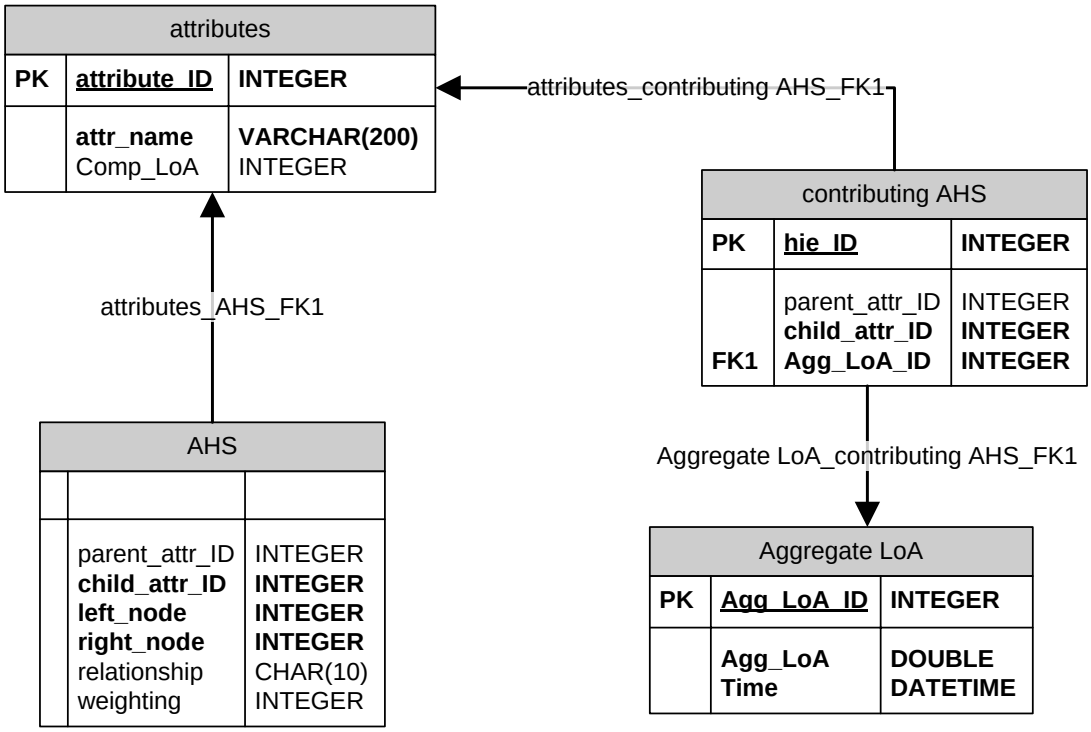

Figure 5.4 LoA-APDB ER Model

5.3.3 LoA-effecting Attributes Collection Module (LoA-ACM)

The LoA-ACM module performs three tasks. Firstly, it interacts with all of the authentication services involved in an authentication instance to receive the SAML assertions that contain the contributing LoA-effecting attributes. Secondly, it parses the assertions to retrieve the contributing LoA-effecting attributes' names. Thirdly, it sends the retrieved data to the ALoA-DM. Figure 5.5 shows the sequences involved

when a user authenticates to a GEA-LoADM model-aware authentication service.

One issue we have observed during the design of the LoA-ACM module is that while conveying the LoA-effecting attributes between different parties (e.g. between different modules in the GEA-LoADM model, or between an authentication service provider and relying party), a unified attributes namespace is required so that both parties can refer to the same attributes without confusion. One way to address this issue is to use SAML authentication context (AC) specification [Samlac]. As we have discussed in Section 3.2, SAML AC specification has defined an XML-based syntax for constructing ACs. Most of the LoA-effecting attributes we have identified are defined in the AC specification (i.e. they have a unified identifier), and for attributes that are not defined in the AC specification, we can extend the definition to accommodate them.

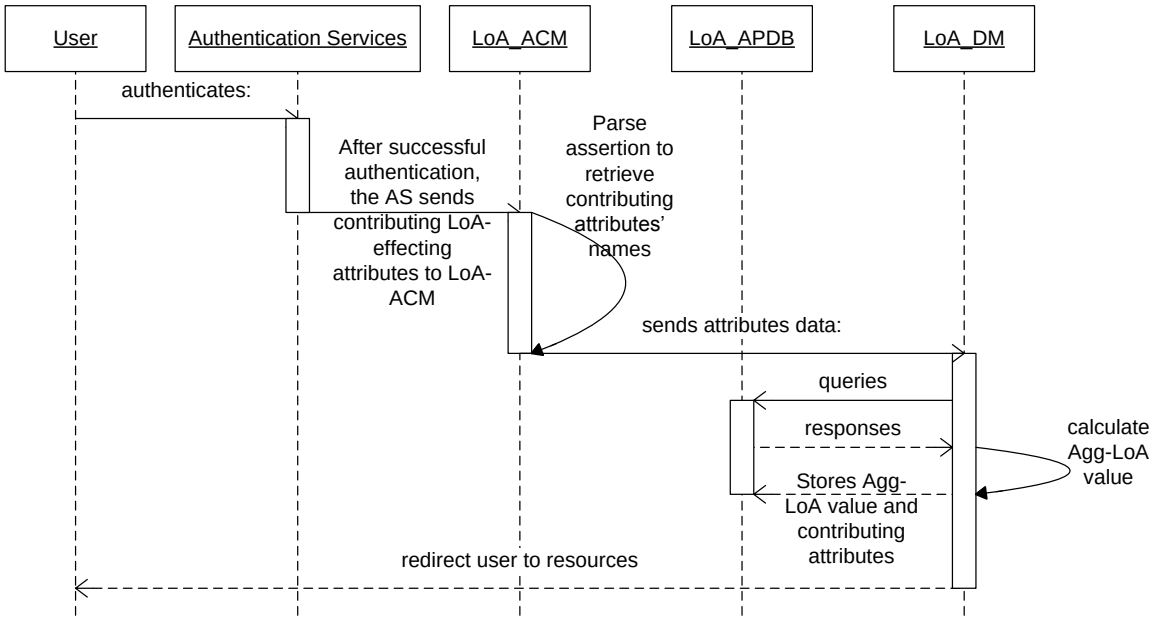

Figure 5.5 GEA-LoADM real-time authentication sequence diagram

5.3.4 Authentication LoA Derivation Module (ALoA-DM)

The ALoA-DM obtains the contributing LoA-effecting attributes names from the LoA-ACM. It then queries the LoA-APDB to fetch the attributes' component LoA values, relationship indicators and weightings and form a sub-LoA-AHS. It then uses the sub-LoA-AHS to derive an aggregated authentication LoA value for the authentication instance. The derivation is done by using the algorithms detailed in

Chapter 4. Once the aggregate LoA value is calculated, the sub-LoA-AHS is stored in the '*Contributing AHS*' table, and the aggregate LoA value, along with a time stamp, is stored in the '*Aggregate LoA*' table in the LoA-APDB for auditing purposes and future reference. Optionally, these data may be stored in a third-party attribute directory managed by an attribute authority for consumption by other relying parties. For example, the data may be sent to the attribute authority in the Shibboleth system for attribute assertion [Shib05, Zhan06], or to the attribute authority for creating and assigning an attribute certificate.

5.4 Model Analysis against Design Requirements

In this section, we analyse the GEA-LoADM model against the design requirements specified in Section 5.1.

R1 - Extensibility:

The model has designed and developed an LoA-AHS structure. By using such a structure, the model can easily support removal of any obsolete attributes, modification of any existing attributes and addition of any new attributes. So if a new authentication service is introduced to the authentication environment, there is no need to alter other attributes in the hierarchy. This feature is particularly important in an area where the technology advances at a rapid pace.

R2 – Support the use of policy-driven attribute LoA values:

The model uses a mapping mechanism to transform existing LoA values to a value between 0 and 1. If another LoA regime is used the only modification is to replace the LoA value mapping mechanism. Other modules in the architecture remain unchanged. In other words, the LoA derivation model will still be applicable regardless of the LoA regime.

R3 – Systematic determination of weightings for LoA-effecting attributes:

Determining the weightings of a set of LoA-effecting attributes in an additive relationship is a challenging issue. The model employs a multi-criteria decision making technique, the analytic hierarchy process (AHP), to establish the weightings of such attributes. By using the pair-wise comparison method, the decision-maker can make subjective decision based on the importance of the services and use AHP to systematically measures the relative weightings between attributes, deriving overall

weightings for all of the attributes.

R4 – Derivation of aggregate LoA:

In an authentication instance, an aggregate LoA can be derived from a set of LoA-effecting attributes based upon their component LoA values. The model has identified a comprehensive set of LoA-effecting attributes, identified two types of relationships for the LoA-effecting attributes and developed three aggregate LoA derivation algorithms. By combining the algorithms and taking the LoA-effecting attributes' component LoA value as input (for attributes that are in an additive relationship, their respective weightings are also required), the model can derive an aggregate LoA for the entire authentication instance.

R5 – LoA conveyance:

The model uses the SAML authentication context specification to convey LoA and LoA-effecting attributes. In this way, all of the attributes and the aggregate LoA are transmitted in a standard, unified and secure manner, thus providing security and convenience for sending and receiving parties.

R6 – Modular structure:

The design of the model uses a modular approach, as described in Section 5.3. Each module of the model is designed to fulfil a specific function and can easily be plugged in or taken out. For example, we can change the aggregate LoA derivation algorithm without modifying/interfering with other components of the model. The LoA-effecting attributes can be added/modified/removed without affecting the attribute collection and aggregate LoA derivation.

R7 – Security and **R8 – Performance**: The security and performance evaluations will be conducted in Chapter 6.

5.6 Chapter Summary

This chapter has given the design details of the GEA-LoADM model, by which the aggregate LoA, as influenced by multiple LoA-effecting attributes, can be systematically estimated. There are two novel contributions in this model design. The first is the LoA-AHS structure, by which a large number of LoA-effecting attributes can be organised into a hierarchical structure with distinctive mutual relationships. The second major contribution is the design of three aggregated LoA derivation

algorithms. They capture the two identified relationships. With the use of these algorithms, supported by the LoA-AHS structure and additional architectural components, the model is able to systematically and automatically derive a composite LoA value given a set of LoA-effecting attributes in an authentication instance. The major advantage of this model is its ability to accommodate a complex set of attributes and to provide a quantitative measure for authentication assurance levels in the face of complex authentication tasks. In the next chapter, we will prototype and evaluate the GEA-LoADM model using a real-life case study. The performance and security of the GEA-LoADM model will also be analysed.

Chapter 6

GEA-LoADM Real System Evaluation

6.1 Chapter Introduction

Chapters 4 and 5 presented the design of three aggregate LoA derivation algorithms and the design of the GEA-LoADM model. This chapter presents a prototype of the GEA-LoADM model. Based on the prototype, the performance of the model is evaluated to further assess the costs introduced as a result of using this approach. A security evaluation is also conducted to assess the resistance level of the model against various security attacks. Section 6.2 reports an implementation of the GEA-LoADM model in detail. The implementation is done in a real-time system, the MAIS (Multi-Agency Electronic Information Sharing) system. Section 6.3 gives a performance evaluation of the GEA-LoADM, and the security evaluation is presented in Section 6.4. Finally, Section 6.5 concludes the chapter.

6.2 GEA-LoADM Model Prototype

6.2.1 Multi-Agency Electronic Information Sharing (MAIS) System

The Pilot Electronic (Web-based) Multi-Agency Information Sharing (MAIS) System for Mentally Disordered Offenders is a project awarded by the NHS Service Delivery and Organisation R&D Programme to the School of Computer Science, in collaboration with the Psychiatry Research Group in the School of Community-Based Medicine, at The University of Manchester. The aim of the project is to establish a multi-agency information sharing IT network that will provide a mechanism through which information about a recently released prisoner, who has severe and enduring mental health problems, can be shared promptly, securely and reliably between the community health (police custody nurses, forensic medical examiners and the criminal justice mental health team) and criminal justice agencies (i.e. police custody sergeants). The MAIS system is a web-based networked system involving the establishment of a secure web-based e-Workbench service to provide browser-based fine-grain controlled access to the health records of mentally ill prisoners upon their

release from the MAIS network prisons.

Figure 6.1 shows the architecture of the MAIS system. From the figure, it can be seen that the e-Workbench is equipped with the following components: a database that contains a selection of the prisoner's personal, criminal and health information; a number of authentication services; a role-based access control service to ensure users from different agencies are only able to access the data they are entitled to; a data access service that facilitates the database access and delivers the required information to the user; and an audit service that records all activities related to the data access in the MAIS system.

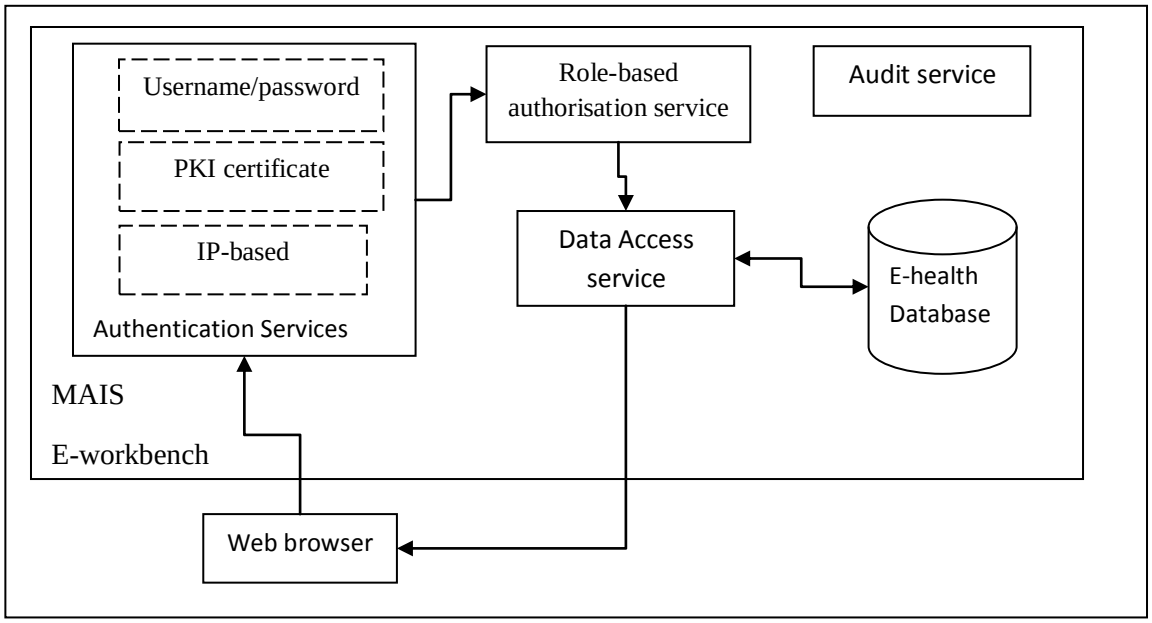

Figure 6.1 MAIS system architecture

At the current stage, the access control service of the MAIS system works as follows. It employs three authentication services: the first one is a username/password-based authentication service, the second one is a PKI certificate-based software/hardware based authentication service and the third one is an IP-based authentication service. An access control process begins with user authentication, in which a user can choose to use either the username/password-based authentication service or the PKI certificate-based authentication service. The IP-based authentication is a compulsory service. More discussions about the compulsory use of IP-based authentication service are given in the next paragraph. In the case of successful authentication, the role-based authorisation service assigns one of two roles (one is 'police' and the other is 'clinician') to the user. This role assignment is done based on the user's ID. Each of

the roles is assigned with a set of access privileges. For example, as shown in the use-case diagram in Figure 6.2, the ‘clinician’ role has ‘write’ access to the prisoner data, whereas the ‘police’ role can only view the data. Another example is that the ‘clinician’ role can access a prisoner’s ‘*prescription detail*’ data, but the ‘police’ role cannot. These access policies are used by the authorisation service to make an access control decision. The decision is executed by the data access service to deliver the appropriate prisoners’ data to the user making the request. During the access control process, the audit service records the user ID, user access time, user’s access IP address and user’s activities, including what data information for which prisoner is viewed/modified by whom at what time. The auditing service ensures accountability in data access and usage.

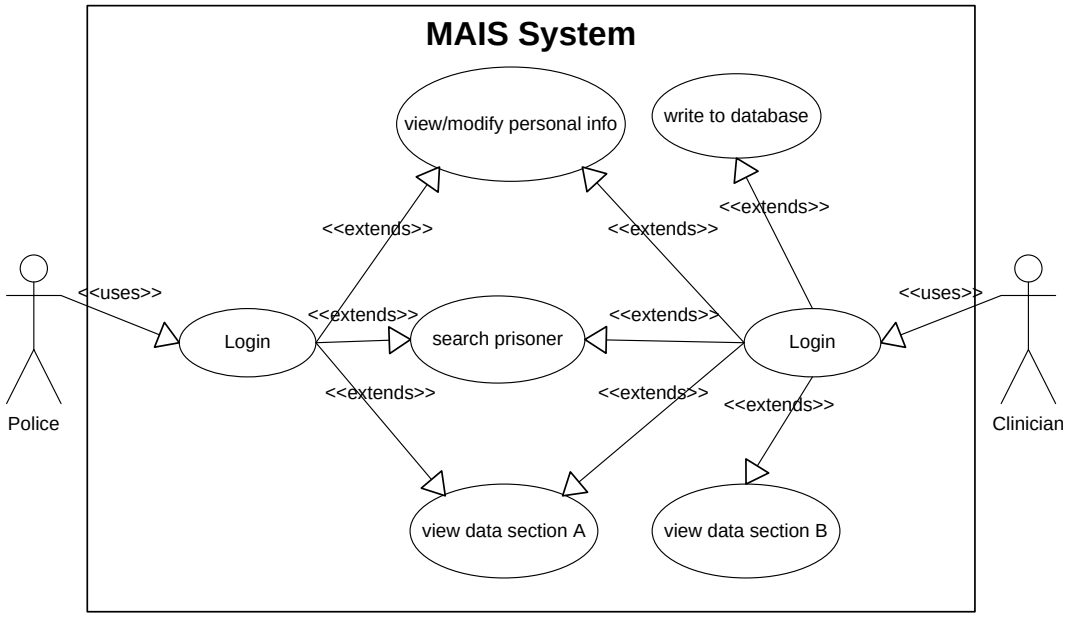

Figure 6.2 MAIS system use case diagram

Owing to the requirement for the project, IP-based authentication is a compulsory authentication service. This means that users can only access data stored in the MAIS system from a few pre-defined IP subnets, such as a police custody suite or a clinician’s office. This security policy was requested by the project stakeholders. It is enforced to provide enhanced protection of the highly sensitive personal information and health data of the prisoners. However, this policy greatly reduced the accessibility of the MAIS system. For example, if a clinician from an agency is called out for an emergency regarding a prisoner’s mental health problems, the clinician would want to check the mental health history and prescription details of the prisoner from his/her

Internet-enabled mobile phone or an Internet-connected computer in the visiting location. However he/she cannot do this because the system is IP-restricted and he/she can only access the MAIS service from his/her office computer.

By integrating the GEA-LoADM model into the MAIS system and linking the authentication LoA to access control decisions in the MAIS authorisation process, we can achieve a more fine-grained access control without sacrificing the usability of the system. For example, there is no need to make compulsory use of the IP-based authentication as the whole point of this security policy is to improve the strength of authentication, thus enhancing the protection of prisoners' data. If a user does not opt for the IP-based authentication, then the user will be assigned with a lower authentication LoA. A lower authentication LoA means less access privilege should be granted to the user. For example, if a user accesses MAIS data using his/her office computer and opts for IP-based authentication, then the user could be granted with 'read', 'write' and 'append' access to the data. However, if the same user accesses the data from somewhere else, and/or does not opt for IP-based authentication, then he/she would only be granted with 'read' and 'append' access. In this way, the restricted user privileges limit the risk of data being compromised and also provide enhanced protection of prisoners' data. The next section presents the implementation and integration of the GEA-LoADM model with the MAIS system.

6.2.2 Integrating GEA-LoADM into the MAIS System

Figure 6.3 shows the integration of the GEA-LoADM model into the MAIS system. Comparing this figure to Figure 6.1, we can see that in order to integrate the GEA-LoADM model, the MAIS system needs to have two modifications. The first modification is to change the authentication services to make them send LoA-effecting attributes to the GEA-LoADM model upon successful authentication. A more detailed discussion of this is given in Section 6.2.2.4. The second modification is to define the LoA requirements of each section of the e-health database and to write access control policies so that they use the LoA as one of the attributes for access control. For prototyping purposes, the author has assigned LoA values to different sections of the e-health database and the authorisation service's security policies have been amended accordingly. In the next four sub-sections, we follow the steps discussed in Section 5.3 to implement the GEA-LoADM model.

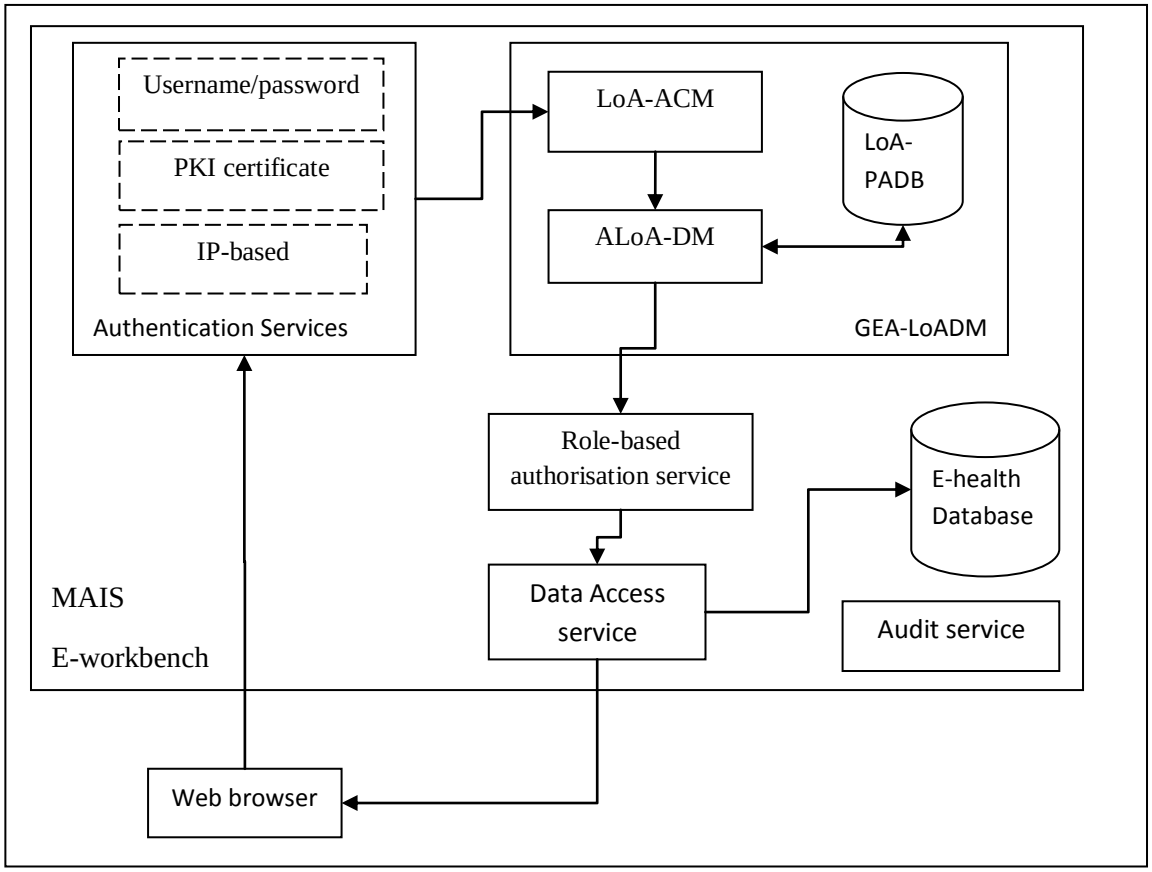

Figure 6.3 MAIS architecture with GEA-LoADM

6.2.2.1Constructing an LoA-AHS for the MAIS system

The first step in building the GEA-LoADM model is to identify the LoA-effecting attributes from the MAIS authentication environment and construct the LoA-AHS structure. As shown in Table 6.1, we have identified seven LoA-effecting attributes based on three authentication services in the MAIS system. The username/password authentication service has '*password*' as the authentication credential and '*SSL*' as the authentication protocol. The certificate-based authentication service has '*X.509 ID credential*' and '*X.509 proxy credential*', and the keystores are '*software token device*' and '*hardware token device*'. The authentication protocol used is '*SSL*'. The IP-based authentication service has no further attributes. In addition, the component LoA values of the attributes are estimated. Some of these values are taken from the NIST e-authentication guideline [Burr06] and some of them (i.e. those not defined by NIST) are assigned by the author for prototyping purposes. The attributes whose LoA values have been given by the author are annotated with an asterisk.

Category	Attributes	Sub-attributes	Component LoA value
MAIS LoA-effecting attributes	Authentication Credentials	• Password	2
		• X.509 ID Credential	4
		• *X.509 Proxy Credential*	*2*
	Authenticator Transport Protocol	• Private key Proof of Procession (PoP), e.g. SSL	4
	Key store	• Software token device	3
		• Hardware token device	4
	Location*	• IP-based authentication	*2*

Table 6.1 MAIS system LoA-effecting attributes list

Based on the LoA-effecting attributes we have identified, we can construct the LoA-AHS structure for the MAIS authentication environment as shown in Figure 6.4.

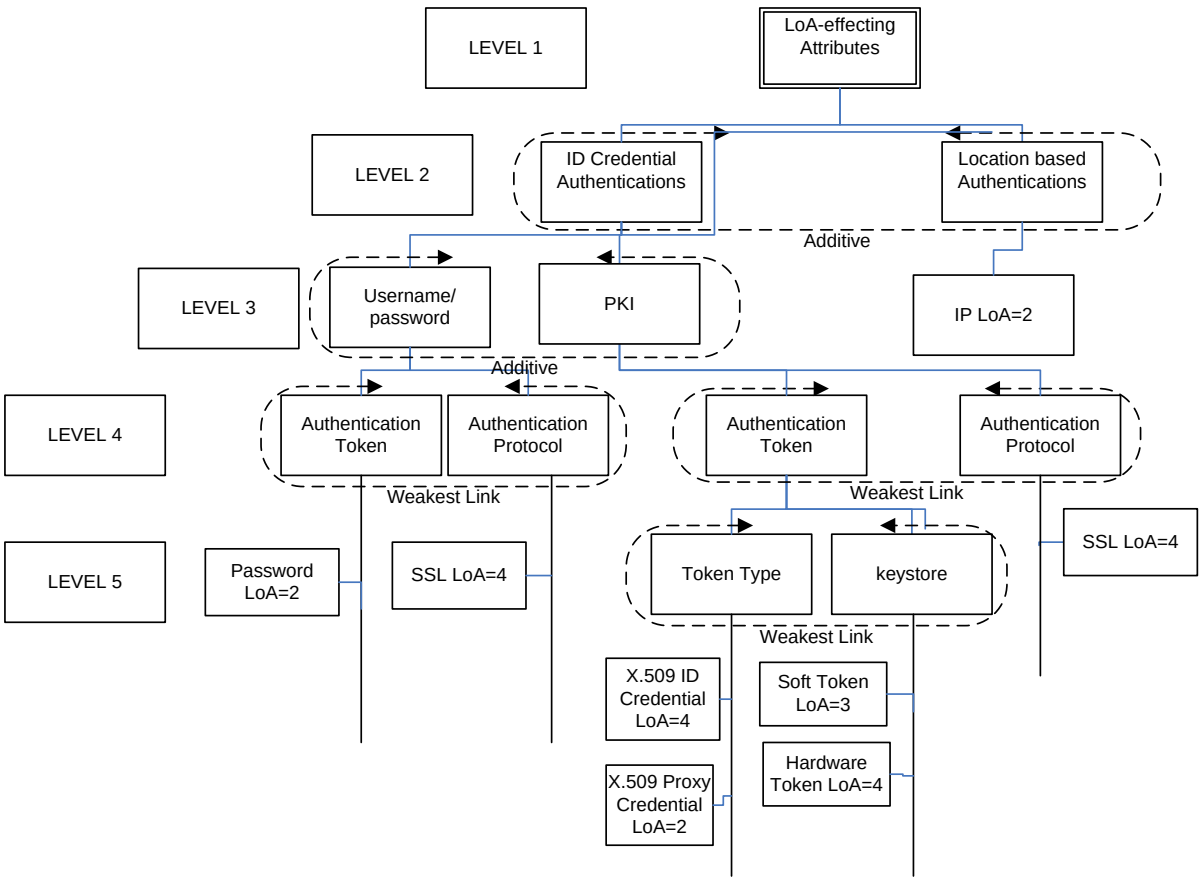

Figure 6.4 MAIS authentication environment LoA-AHS structure

6.2.2.2 Estimating weightings of additive attributes

After structuring all of the LoA-effecting attributes and identifying the relationships for each group of attributes, the weightings of the additive attribute groups need to be determined. We follow the steps described in Section 4.4.1 to determine the weightings of the additive attributes in the MAIS LoA-AHS structure. For prototyping purposes, the author has chosen pair-wise comparison values for the additive attributes groups, as shown in Table 6.2 and 6.3. A Java web application is developed to calculate the weightings for each of the additives attributes. Figure 6.5 shows the GUI of the application. In Table 6.2 and 6.3, the pair-wise comparison matrix, the calculated eigenvector and the corresponding normalised weightings of each additive attributes groups are presented.

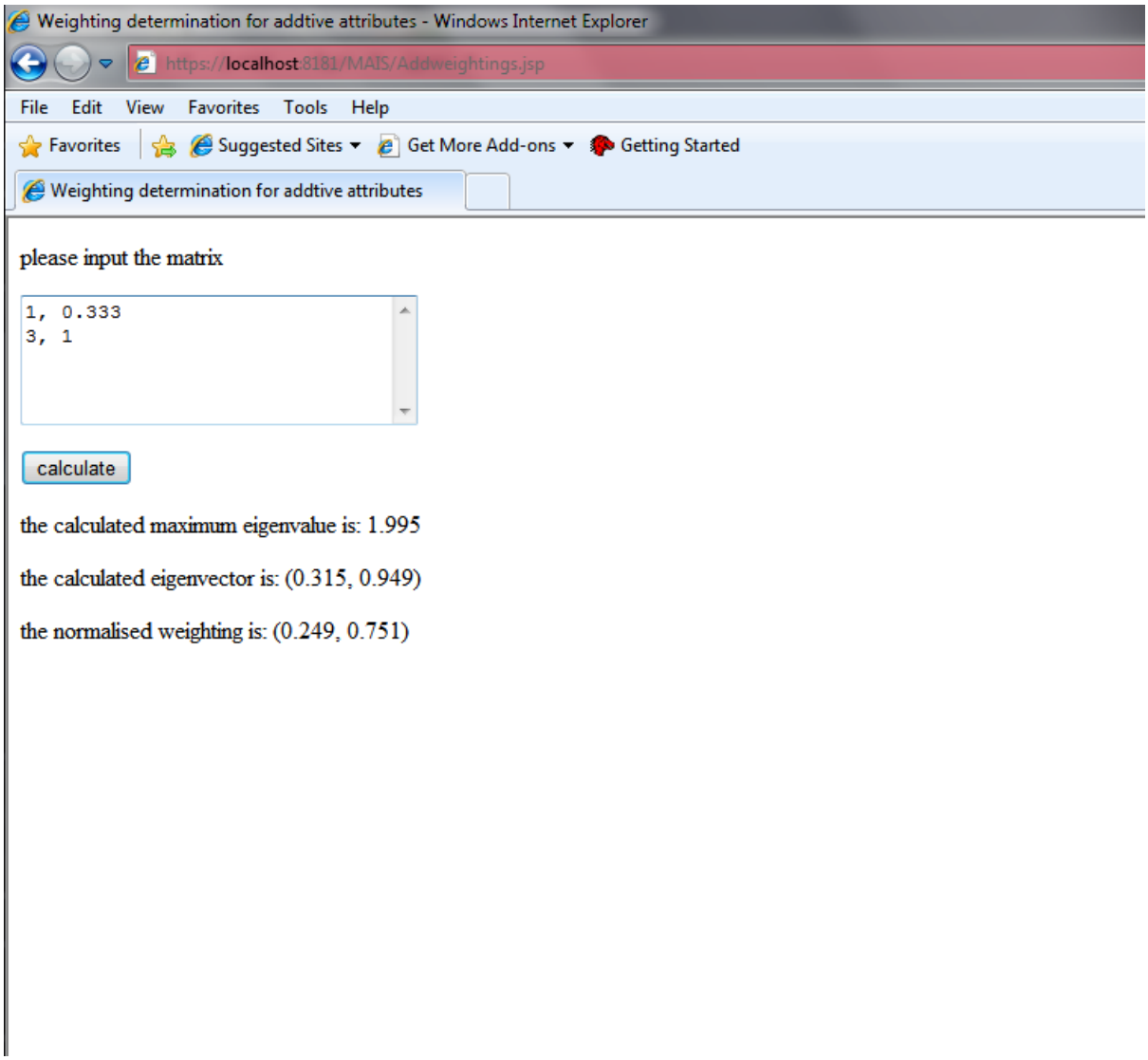

Figure 6.5 Weighting determination GUI

	ID credential-based Authentication	Location-based Authentication	Eigenvector	Normalised weightings
ID credential-based Authentication	1	4/1	4	0.8
Location-based Authentication	1/4	1	1	0.2

Table 6.2 Pair-wise comparison matrix for level 2 additive attributes group

	Username/password Authentication	Certificate-based authentication	Eigenvector	Normalised weightings
Username/password Authentication	1	1/3	0.315	0.249
Certificate-based Authentication	3	1	0.949	0.751

Table 6.3 Pair-wise comparison matrix for level 3 additive attributes group

At this point, all of the LoA-effecting attributes in the MAIS authentication environment have been structured, their component LoA values estimated and the weightings of additive attributes calculated. The next step is to implement and populate the LoA-APDB database and store the information into the database for the real-time component of the GEA-LoADM to use.

6.2.2.3 Implementation of the LoA-APDB

The LoA-APDB database is implemented using Mysql technology. As mentioned in Section 5.4.2, four tables have been created in the LoA-APDB database. Appendix B contains the database schema.

6.2.2.4 Implementation of the LoA-ACM module

The LoA-ACM module is developed as a Java servlet that does the following tasks:

1). Receiving SAML assertions that contain the contributing LoA-effecting attributes' names from authentication services during the authentication process in an authentication instance.

2). Parsing the assertions and retrieving the LoA-effecting attributes' names from them.

3). Sending the retrieved information to the ALoA-DM.

Here we use an example to illustrate the process of the LoA-ACM module. Assume a user, Bob, chooses to authenticate to the MAIS system using both username/password and IP-based authentication services. After successful authentication, the LoA-ACM module receives one SAML assertion from each of the authentication services, as shown in Tables 6.4 and 6.5. The LoA-ACM parses the SAML assertions and retrieves the corresponding LoA-effecting attributes' names. Table 6.6 shows the retrieved LoA-effecting attributes' names from the assertions. This information is then sent to the ALoA-DM module via an SSL channel to ensure the integrity and the confidentiality of the message.

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="db3c2838db744f704c5363059bb7a0eb" IssueInstant="2010-05-
17T14:58:34.763Z" Issuer="EGA-LoADM" MajorVersion="1"
MinorVersion="1">
      <Conditions>
            <DoNotCacheCondition></DoNotCacheCondition>
      </Conditions>
      <AttributeStatement>
            <Subject>
                  <NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
NameQualifier="EGA-LoADM:AS">password authentication</NameIdentifier>
                  <SubjectConfirmation>

      <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sender-
vouches</ConfirmationMethod>
                  </SubjectConfirmation>
            </Subject>
            <Attribute AttributeName="password authentication token"
AttributeNamespace="urn:bea:security:saml:AC">
                  <AttributeValue>password</AttributeValue>
            </Attribute>
            <Attribute AttributeName="password authentication
protocol" AttributeNamespace="urn:bea:security:saml:AC">
                  <AttributeValue>ssl</AttributeValue>
            </Attribute>
      </AttributeStatement>
</Assertion>
```

Table 6.4 Username/password authentication service SAML assertion

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="db3c2838db744f704c5363059bb7a0eb" IssueInstant="2010-05-
17T14:58:34.763Z" Issuer="EGA-LoADM" MajorVersion="1"
```

```
MinorVersion="1">
      <Conditions>
            <DoNotCacheCondition></DoNotCacheCondition>
      </Conditions>
      <AttributeStatement>
            <Subject>
                  <NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
NameQualifier="EGA-LoADM:AS">IP authentication</NameIdentifier>
                  <SubjectConfirmation>

      <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sender-
vouches</ConfirmationMethod>
                  </SubjectConfirmation>
            </Subject>
</Assertion>
```

Table 6.5 IP-based authentication service SAML assertion

Level 3 attributes	**Level 4 attributes**	**Attributes values**
Password authentication	Password authentication token	password
	Password authentication protocol	SSL
IP authentication		

Table 6.6 Retrieved LoA-effecting attributes names from SAML assertions

6.2.2.5 Implementation of the ALoA-DM module

The ALoA-DM module receives the contributing LoA-effecting attributes' names from the LoA-ACM and queries to the LoA-APDB to retrieve both the contributing attributes' and their parent attributes' names (the database stores the hierarchical structure), component LoA values, weightings and relationship indicators. These attributes form a sub-LoA-AHS, as illustrated in Figure 6.5.

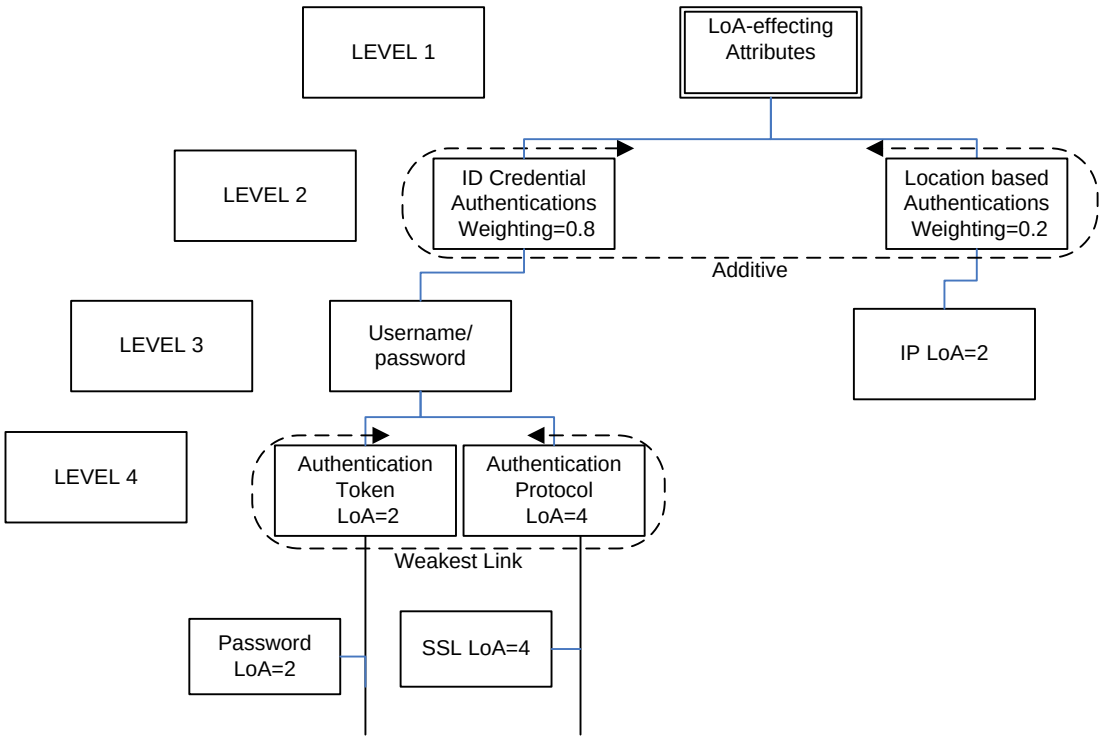

Figure 6.6 The sub-LoA-AHS for an authentication instance

The ALoA-DM uses the structure as input to calculate the final aggregate LoA value. Using the algorithms described in Chapter 4, the results are the following:

Aggregate LoA value for this authentication instance	Use $ALoA_{AD}$-SL algorithm	Use $ALoA_{AD}$-PT algorithm
	0.71	0.77
NIST LoA definition	3	4

Table 6.7 Aggregate LoA calculated in two additive algorithms

The $ALoA_{AD}$-SL algorithm gives an aggregate LoA value '0.71'; the $ALoA_{AD}$-PT algorithm gives an aggregate LoA value '0.77'; and based on the mapping mechanism discussed in Section 4.4, the threshold of NIST level 3 and 4 is 0.75, thus the aggregate LoA values are transformed to 3 and 4 in the NIST definition respectively. It can been seen that the small changes in the numerical values of the LoA can lead to the same procedure being placed in two different NIST LoA levels. An observation can be made from this result. I.e. the numerical LoA values represent the authentication assurance in a more precise way than NIST LoA levels. More refinement for NIST LoA levels is needed.

After the derivation of the aggregated LoA, the corresponding LoA-effecting

attributes and the LoA value for this authentication instance are stored in the LoA-APDB for authorisation services to use.

6.3 GEA-LoADM Performance Evaluation

This section aims to investigate the level of the additional overheads the GEA-LoADM introduces into an authentication process. In the following experiments, the Authentication Execution Times (AETs) in both the GEA-LoADM-enabled MAIS system and the MAIS system without the GEA-LoADM are measured and compared. An AET is defined as the time difference between when an authentication service receives an authentication request and when the authentication service makes an authentication decision for the request. The unit of the AET is a millisecond.

Two sets of experiments (Set A and Set B) have been conducted for the evaluation. Set A is performed in the GEA-LoADM enabled MAIS system and Set B is performed in the MAIS system without the GEA-LoADM. Each set contains four identical experiments. Experiment one uses a username/password authentication service. Experiment two uses a certificated-based authentication service. Experiment three uses username/password and IP-based authentication services, and experiment four uses certificate-based and IP-based authentication services.

Prior to performing the experiments, two main results were anticipated. Firstly, that all the AETs in Set A experiments would be longer than the AETs in Set B experiments. The reason is apparent as the GEA-LoADM would add additional overheads to the MAIS system. Secondly, the AETs in both sets would be longer when multiple authentication services are used (i.e. experiments 3 and 4) than the AETs when a single authentication service is used (experiments 1 and 2). This is because more computational time will be consumed when the sizes of attributes' input for the algorithms are larger, as we have evaluated in Section 4.5.2.

6.3.1 Evaluation Environment

The experiments are hosted on a Windows 7 OS running on a ThinkPad laptop with two 2.53GHz processors and 3072MB of memory. The prototype is implemented as a JAVA web application with Java EE™ Runtime Environment version 5. The performance evaluation tool used is NetBeans IDE v6.8.

6.3.2 Experiment Results

Figure 6.7 shows the results of the experiments. As can be seen in the figure, one set is from the MAIS system with the GEA-LoADM and the other set from the MAIS system without the GEA-LoADM. The four experiments are 'username/password authentication service', 'certificated-based authentication service', 'username/password and IP-based authentication services' and 'certificate-based and IP-based authentication services', respectively. Two observations can be made from these results. The first one is that the AETs in Set A are significantly bigger than the AETs in Set B. This observation is in line with our expectations. The average difference is around 4 times longer. For example, in these experiments, the average AET in Set A is 2.3 seconds, but the average value in Set B is 0.64 seconds. Another observation is that in Set A, the AETs in experiments 3 and 4 are much longer than in experiments 1 and 2. This result is also expected.

In order to improve the performance and reduce the AET in the GEA-LoADM model, we need to find out which component in the GEA-LoADM cost the most AET. For this purpose, we have done another set of experiments and investigated the execution times of each of the components involved in the authentication process. Figure 6.8 shows the experiment results. It shows three sets of execution times in milliseconds against the four experiments. The three sets are from the LoA-ACM module, the ALoA-DM module and the authentication service, respectively. The four experiments are the same as the previous ones. The results show that the LoA-ACM module contributes the most AETs. After further investigating the program, we found that this is because the SAML assertions are XML-based messages, and parsing these messages (typically involving the construction of a DOM tree) is a very time-consuming operation. Therefore, it costs a large amount of execution time. There is little we can do from the software point of view, but a high-performance computer would be able to reduce this execution time.

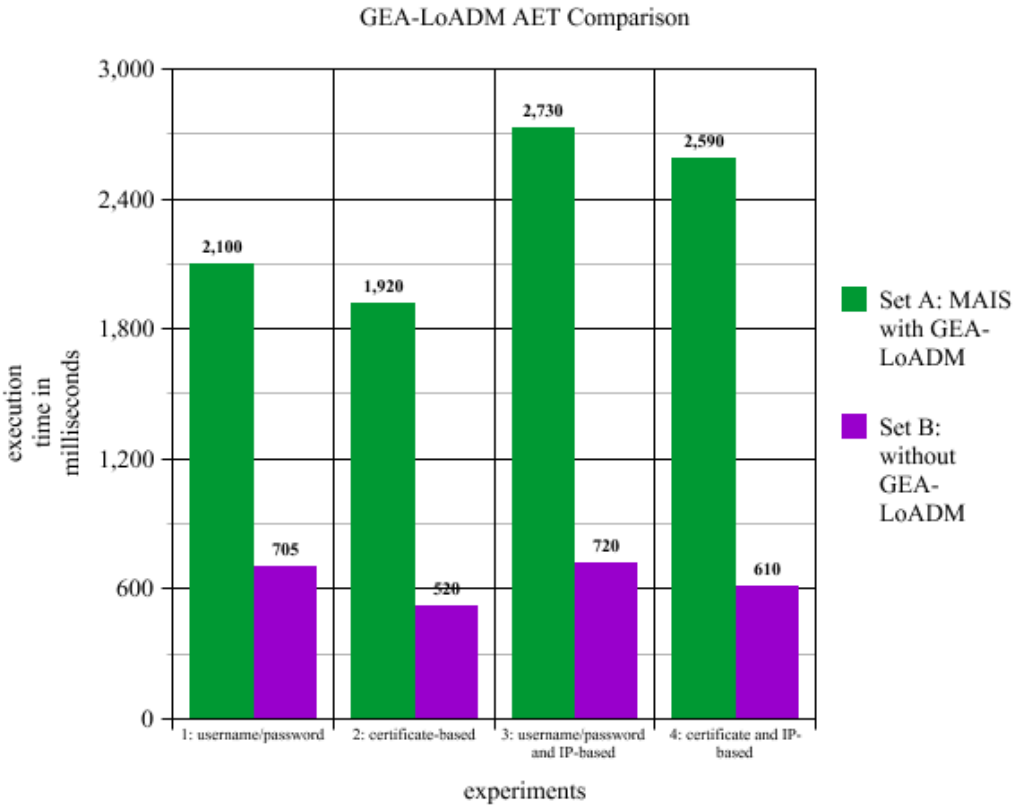

Figure 6.7 GEA-LoADM AET experiments

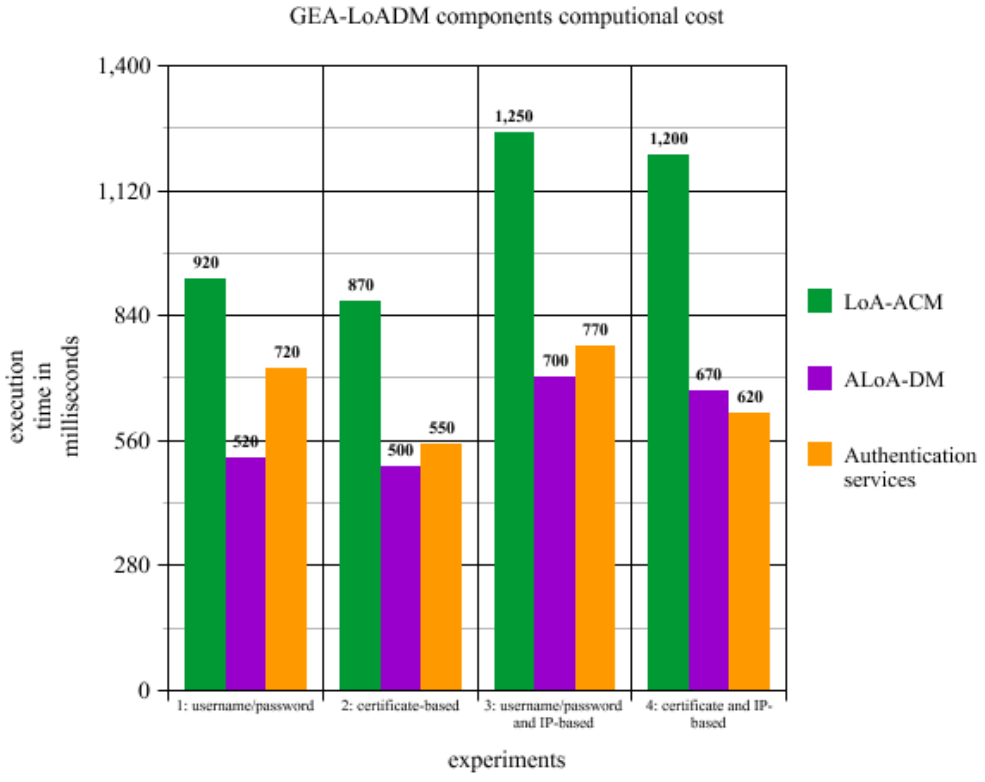

Figure 6.8 GEA-LoADM components execution time experiments

6.4 Tests against Security Attacks

This section evaluates the security strength of the GEA-LoADM and discusses whether the overall security level of the MAIS system would be weakened by applying the GEA-LoADM model.

From the discussion in Section 6.2.2, we can see that the GEA-LoADM model does not interfere with the user-to-system authentication process; therefore, the security of this part is unchanged. However, because the authentication services send SAML assertions to the GEA-LoADM, the authenticity of the authentication services and the integrity and confidentiality of the SAML assertions need to be guaranteed. We do so by issuing certificates to each of the authentication services and enabling SSL mutual authentication services to verify the authenticity of the authentication services.

We have attempted some well-known attacks against our prototype and analysed the resistance level of the GEA-LoADM against them. Four security attacks have been attempted against the system: the first one is a forged assertion attack, the second one is a replay attack, the third one is a man-in-the-middle attack and the fourth one is a denial of service (DoS) attack. In the forged assertion attack, an attacker forges or alters an SAML assertion and sends it to the GEA-LoADM model. Based on the countermeasure specified in the SAML specification [Samlv2], the authentication service can digitally sign the SAML assertion before sending it to the GEA-LoADM model, thus providing both message integrity and authenticity. The replay attack is when an attacker intercepts a valid data transmission and retransmits it to the GEA-LoADM. The countermeasure is to use the SSL transmitting channel between the authentication services and the LoA-ACM, thereby preventing the message's eavesdropping and interception. In addition, we have enabled a timestamp in the signature on an assertion (alternatively, a nonce continued by the receiving module can be included) to provide an additional assurance. The man-in-the-middle attack is a form of active eavesdropping in which the attacker makes independent connections with the authentication services and the GEA-LoADM model and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. Again, the countermeasure is to use SSL mutual authentication to verify the authenticity of the authentication services thus ensuring the authenticity of the

message. In a DoS attack, an attacker attempts to prevent legitimate users from accessing services/resources by making them unavailable. The most common form is to send enormous valid but useless requests to a service provider thus blocking the network and/or crashing the server. Since the operations in the GEA-LoADM model are computationally more expensive than conventional authentication services as we demonstrated in the last section, the GEA-LoADM model could potentially be vulnerable to this type of attack. However, as the GEA-LoADM model takes messages sent by authentication services rather than receiving authentication requests directly from users, we can thwart this attack by assuring the authenticity between the authentication service and the GEA-LoADM using SSL mutual authentication. In this way, the GEA-LoADM could filter out invalid messages more efficiently. They will have to first successfully pass the authentication services. In other words, only authorised users could cause harm by launching such attacks; measures such as auditing can be used to identify and deter the attackers.

In addition, the protection of the LoA-APDB database is also an important issue since the aggregate LoA values of each authentication instance (live or completed) are stored there. The database should be installed on a separate machine where only the server has access to it, and the database machine should not be linked to the Internet directly.

6.5 Chapter Summary

This chapter has described a prototype of the GEA-LoADM model based on a real-life system, the MAIS system. Using the prototype, performance and security evaluations have been conducted. Major contributions demonstrated in this chapter are threefold. Firstly, the GEA-LoADM model has been implemented in a real-life system. Secondly, based on this prototype, the performance of the model has been investigated and compared with a conventional authentication process to understand the extent of the overheads introduced by the model. Thirdly, by simulating well-known attacks on the prototype, the security of the model has been validated, which has demonstrated that the model is able to resist these attacks.

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

Chapter 7

Conclusion and Future Work

This chapter summarises the work presented in this thesis, gives the conclusions drawn from the research findings, and finally, recommends future work.

7.1 Thesis Summary

7.1.1 Review of the Thesis

The work presented in this thesis can be arranged into four parts: research background, authentication levels of assurance (LoA) derivation algorithms, the generic e-authentication levels of assurance derivation model (GEA-LoADM), and the implementation and evaluation of the model.

Research Background

The thesis has explained what authentication LoA is and how to link it to a user's access privilege to achieve a fine-grained access control. Chapter 2 gave an overview of the authentication LoA and investigated its related works. It also provided an in-depth survey of e-authentication technologies and examined the authentication solutions in large-scale distributed system environments. Chapter 3 identified the LoA-effecting attributes from different authentication LoA-related works. It further identified two types of relationships from them, namely the weakest link relationship and the additive relationship. Finally it investigated potential methods that could be used to quantify the impact (weighting) of additive relationship attributes on an aggregate LoA.

LoA Derivation Algorithms

Chapter 4 presented our innovative authentication LoA derivation algorithms. It first specified the requirements for designing such algorithms. It then presented the design of one LoA derivation algorithm for attributes with the weakest link relationship, and the design of two LoA derivation algorithms for attributes with the additive relationship. Finally, it evaluated the performance and results of the algorithms.

GEA-LoADM Model

Chapter 5 presented the design of a real-time authentication LoA derivation model, the GEA-LoADM, and integrated the LoA derivation algorithms into the model. It first detailed the design requirements for designing such a model. It then described the architecture and architectural components of the model. Finally, it analysed the model against its design requirements.

Implementation and Evaluation

The implementation of the GEA-LoADM model was done on a real-life system, the MAIS system. Based on the prototype, the performance of the model has been evaluated and compared to the system without the GEA-LoADM. This evaluation and comparison further assess the costs introduced as a result of using this approach. A security evaluation has also been conducted to assess the resistance level of the model against well-known security attacks.

7.1.2 Contributions

The thesis has made the following contributions and discoveries:

The LoA-effecting attributes hierarchical structure (LoA-AHS)

The author has identified and analysed a wide range of LoA-effecting attributes, examined the mutual relationships among them and designed a novel LoA-effecting attributes hierarchical structure (LoA-AHS). The LoA-AHS is flexible and extensible, and can accommodate existing, as well as future emerging, LoA-effecting attributes. By grouping multiple attributes into hierarchical levels and different categories based on their mutual relationships, this structured approach provides a systematic method for determination of an aggregated LoA value for an authentication instance, given component LoA values of the attributes involved as well as the weightings of the additive attributes.

Aggregate LoA Derivation Algorithms

The second major novel contribution is the design, prototype and evaluation of the three aggregate LoA derivation algorithms. One of the algorithms is for capturing the weakest link relationship among attributes and the other two are for capturing the additive relationship among attributes.

Generic e-authentication LoA derivation model

The third major novel contribution is the design of the GEA-LoADM model that incorporates the aggregate LoA algorithms and the AHS structure. The model is capable of gathering LoA-effecting attributes, retrieving component LoA values of these attributes and the weightings of those attributes that are in an additive relationship, and deriving an aggregate LoA value for an authentication instance in real-time.

Prototype-based evaluation

A prototype-based evaluation has been carried out, which demonstrates that our solution, the GEA-LoADM, is capable of deriving an aggregate LoA based on a set of LoA-effecting attributes in real-time. It also demonstrates that the model is secure. The overheads introduced by the GEA-LoADM solution are the price to pay in order to achieve a more fine-grained access control. Real-system experiments have shown that, on average, the GEA-LoADM increases an authentication delay by 3.6 times in comparison to the system when LoA derivation is not enabled. A bigger portion of this increase is caused by parsing the XML messages generated as a result of using SAML assertions.

To summarise, the research has explored the use of authentication LoA to facilitate a fine-grained access control that allows a user's access privilege to be linked to the authentication LoA in identifying the user.

7.2 Future Work

The following are recommended for future work.

- The functionality of the GEA-LoADM model can be extended. For example, it can be equipped with a negotiation mechanism so that if the authentication LoA of a user is not sufficient for certain resources/services access, instead of denying the access request, the model can negotiate with the user to allow the user to choose an alternative authentication method in order to achieve the desired authentication LoA, or to grant the user with a lower level of access privilege.
- The current prototype uses SAML assertions as the underlying LoA-effecting attributes mechanism to encapsulate and transport their component values, and other data required by the GEA-LoADM. As shown in our experiments that XML

message parsing is a time-consuming task, other transportation methods should be investigated to find a more efficient way of conveying data between authentication services and the GEA-LoADM model.

- The evaluation of the GEA-LoADM has only considered the user-to-system scenario. More work is required to evaluate the model under other authentication scenarios, such as the assertion-based authentication scenario and delegation scenario. The GEA-LoADM model has considered the credential delegation, and more work is required to prototype the solution in a Grid application context to demonstrate the applicability of the model.
- A ubiquitous computing environment involves the use of more LoA-effecting attributes than a conventional Internet environment. It would be interesting to investigate the use of the GEA-LoADM in a ubiquitous environment. In addition to identifying new attribute types, new authentication LoA derivation algorithms may need to be designed if new relationships among the attributes are identified.

BIBLIOGRAPHY

[Austaf] 'The Australian Access Federation', available at http://www.aaf.edu.au/ last access Oct 200

[Auegov] 'Australian e-Government & Information Management', available at http://www.finance.gov.au/e-government/index.html last access Oct 2009

[Azum93] R. Azuma. 'Tracking Requirements for Augmented Reality', Communications of the ACM, 36(7):50-51, July 1993.

[Burr06] W. E. Burr; D,F. Dodson and W.T. Polk, 'Electronic Authentication Guideline, version 1.02', NIST Special Publication 800-63, April 2006. Available at: http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf last access April 2010

[Caegov] 'Canadian e-authentication', 2004 Available at: http://www.ic.gc.ca/epic/site/ecicceac.nsf/en/h gv00090e.html last access Oct 2009

[Capims] 'The Cryptography API', available at http://msdn.microsoft.com/en-us/library/aa380256%28VS.85%29.aspx last access Oct 2009

[Chae04] J. Chae; G.K.Mostefaoui and Mokdong Chung, 'An Adaptive Security Model for Heterogeneous Networks Using MAUT and Simple Heuristics', Computational Science and Its Applications; LNCS 3046, 2004 pp .983–993,2004

[Conv04] M. J. Covington; M. Ahamad; I. Essa and H. Venkateswaran 'Parameterized Authentication', European Symposium on Research in Computer Security 2004, pp. 276–292, Sophia Antipolis, French Riviera, France 2004.

[Cree04] S. Creese; M. Goldsmith; B. Roscoe and I. Zakiuddin, Authentication for Pervasive Computing, in Proceedings of the First International Conference on Security in Pervasive Computing, 2003, pp.116-129, Boppard, Germany, 2004.

[Euegov] IDABC, Interoperable Delivery of European e-Government Services to public Administrations, Businesses and Citizens, http://ec.europa.eu/idabc/en/home.

[Fips01] National Institute for Standards and Technology, FIPS PUB 140-2: Security Requirements for Cryptographic Modules, http://csrc.nist.gov/cryptval/140-2.htm last access Feb 2010

[Fost03] I. Foster; V. Welch; S. Tuecke; L. Pearlman and C. Kesselman, 'The Community Authorization Service: Status and future', In the Proceedings of 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03), La Jolla, California, 24-28 Mar 2003, pp TUBT003

[Gang01] G.R. Ganger, 'Authentication Confidences', hotos, pp.0169, Eighth Workshop on Hot Topics in Operating Systems, 2001

[Gass05] S. I. Gass, 'Model World: The Great Debate - MAUT Versus AHP', Interfaces Volume 35 , Issue 4 (July-August 2005) Pages: 308-312 Year of Publication: 2005 ISSN:0092-2102

[Hart94] A Harter and A Hopper. 'A Distributed Location System for the Active Office', IEEE Network, January 1994.

[Ietf4120] The Kerberos Network Authentication Service (V5) http://www.ietf.org/rfc/rfc4120.txt last accessed Oct 2009

[IGTF-1] Profile for Traditional X.509 Public key Certification Authorities with Secured infrastructure, version 4; available at http://eugridpma.org/guidelines/IGTF-AP-classic-20050930-4-0.doc. last access Feb 2010

[IGTF-2] Profile for Short Lived Credential Services X.509 Public key Certification Authorities with Secured infrastructure, version 1; available at http://www.tagpma.org/files/IGTF-AP-SLCS.doc. last access Feb 2010

[Incomm] 'InCommon Bronze and Silver Credential Assessment Profiles v0.3', available at http://www.incommonfederation.org/docs/drafts/InC_Bronze_CAP_0.3.doc last access Feb 2010

[Ismcdm] 'International Society on Multiple Criteria Decision Making', available at

http://www.mit.jyu.fi/MCDM/intro.html last access June 2009

[Javaapi] JavaTM 2 Platform Standard Edition 5.0 API Specification available at http://java.sun.com/j2se/1.5.0/docs/api/ last access May 2010

[Josa00] A. Josang and V.A. Bondi, 'Legal Reasoning with Subjective Logic'. Artificial Intelligence and Law, 8(4), pp.289-315, Kluwer 2000.

[Jpegov] Japan, An overview of International Initiatives in the field of Electronic Authentication, 2005 Available at: http://www.japanpkiforum.jp/shiryou/e-auth policy/overview e-auth v07.pdf last access Oct 2009

[Kang01] J. Kangas; A. Kangas; P. Leskinen and J. Pykalainen. 'MCDM methods in strategic planning of forestry on state-owned lands in Finland: applications and experiences'. Journal of Multi-Criteria Decision Analysis 2001;10:257 – 71.

[Keen93] R.L. Keeney and H. Raiffa. Decision with Multiple Objectives: Preference and Value Tradeoffs. Cambridge University Press, New York, 1993.

[Kike05] G.A Kiker; T.S. Bridges; A. Varghese; P.T. Seager and I. Linkov, 'Application of multicriteria decision analysis in environmental decision making', Integration. Environment. Assess. Manage 2005. 1(2), 95–108.

[Korp04] J.Korpela; A.Lehmusvaara; K.Kylaheiko and M.Tuominen, 'Adjusting Safety Stock Requirements with an AHP-based Risk Analysis', System Sciences, conference on Proceedings of the 36th Annual Hawaii international ISBN: 0-7695-1874-5, 2004

[Marc03] J. Marchesini; S.W. Smith and M.Y Zhao, 'Keyjacking: Risks of the Current Client-side Infrastructure', In 2nd Annual PKI Resarch Workshop. NIST, April 2003

[Mess00] T. S. Messerges, 'Using Second-Order Power Analysis to Attack DPA Resistant Software', Lecture Notes in Computer Science, 2000, Volume 1965/2000, 27-78, DOI: 10.1007/3-540-44499-8_19

[Need78] R. M. Needham and M. D. Schroeder, 'Using Encryption for Authentication inLarge Networks of Computers', Communications of the ACM, Vol. 21(12), 1978 pp.993-999

[Nena06] A. Nenadic; N. Zhang; J. Chin and C. Goble, ‘Fame: Adding Multi-Level Authentication to Shibboleth’, IEEE Conference of E-Science and Grid Computing, Page(s):157 - 157, Amsterdam, Holland, 2006.

[Novo01] J. Novotny; S. Tuecke and V. Welch, ‘An Online Credential Repository for the Grid: MyProxy’, hpdc, p. 0104, 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01), 2001.

[OMB03] OMB Memorandum M-04-04, E-Authentication Guidance for Federal agencies, December 16, 2003. Available at: http://www.whitehouse.gov/OMB/memoranda/fy04/m04-04.pdf, last access March 2010

[Odpm04] ODPM (Office of the Deputy Prime Minister). DLTR multi-criteria decision analysis manual; 2004. Available at http://www.odpm.gov.uk/stellent/groups/odpm_about/documents/page/odpm_about_608524-02.hcsp.

[Pkcs11] PKCS #11: Cryptographic Token Interface Standard http://www.rsa.com/rsalabs/node.asp?id=2133 last access Oct 2009

[Rfc3280] R. Housley, et al; “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, available at: http://www.faqs.org/rfcs/rfc3280.html, last access August 2009.

[Rfc3820] S. Tuecke, et al; Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, available at: http://www.faqs.org/rfcs/rfc3820.html, last access July 2009.

[Royb05] B. Roy, ‘Multiple Criteria Decision Analysis: State of the Art Surveys’, ISSN0884-8289, Springer New York, 2005

[Samlv2] Security Assertion Markup Language version 2 specificaitons, http://saml.xml.org/saml-specifications last access Feb 2010

[Samlac] SAML 2.0 Authentication Context specification Available via OASIS. http://docs.oasisopen.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf last access April 2010

[Saat77] T.L. Saaty, ‘Scaling method for priorities in hierarchical structures’, Journal of Mathematical Psychology 15/3 (1977) 234-281.

[Saat80] T.L. Saaty, ‘Multicriteria Decision Making: The Analytic Hierarchy Process’, 1988; Revised and published by the author; Original version published by McGraw-Hill, New York, 1980.

[Saat82] T.L. Saaty, ‘Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World’, RWS Publications, Pittsburgh, PA, 1986; Original version published by Lifetime Learning Publications, 1982.

[Saat90] T. L.Saaty, ‘How to make a decision: The analytic hierarchy process’, European Journal of Operational Research. No: IC/1990/48, pp. 9-26, 1990

[Shib05] Shibboleth Architecture technical overview, 2005, available at: http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf, last access Nov 2006

[Szegov] Switch Pilot Assurance Levels Definition, https://aai-wiki.switch.ch/bin/view/AAIHomeOrgs/AssuranceLevels last access Oct 2009

[Tlsv10] the Transport Layer Security (TLS) protocol, available http://www.ietf.org/rfc/rfc2246.txt last access Oct 2009

[Toda03] P. Todaro, “An Overview of the Kerberos Authentication Protocol”, 2003. http://www.giac.org/practical/GSEC/Pam_Todaro_GSEC.pdf last accessed August 2009

[UKloa00] e-government authentication framework, Version 1.0, December 2000. Available at: http://archive.cabinetoffice.gov.uk/e-envoy/resources-pdfs/$file/authentic.pdf last access March 2010.

[UKloa02] Registration and Authentication e-Government Strategy Framework Policy and Guidelines, Version 3.0, September 2002. Available at: http://archive.cabinetoffice.gov.uk/e-envoy/frameworks-authentication/$file/Registration-AuthenticationV3.pdf last access March 2010

[W3cscw] W3C Web Security Context Working Group http://www.w3.org/2006/WSC/ last access June 2009

[Want92] R.Want; A. Hopper; V. Falcao and J. Gibbons, ‘The Active Badge Location System’, ACM Transactions on Information Systems, Vol. 10, No. 1,

January 1992. 91-102.

[Welc05] V. Welch, ‘Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective’, 2005. Available at: http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf, last access July 2009

[Wssv1.1] Web Services Security specification version 1.1 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss last access Oct 2010

[Xmlenc] D. Eastlake, et al, XML Encryption Syntax and Processing, W3C Recommendation, 10 December 2002.Available at: http://www.w3.org/TR/xmlenc-core/, last access June 2009

[Xmlsig] D. Eastlake, et al, XML Signature Syntax and Processing, W3C Recommendation, 12 February 2002. Available at: http://www.w3.org/TR/xmldsig-core/, last access June 2009

[Yoec02] C. Yoe, ‘Trade-Off Analysis Planning and Procedures Guidebook’. Prepared for Institute for Water Resources. U.S. 2002 Army Corps of Engineers.

[Zhan06] N. Zhang; L. Yao; A. Nenaic; J. Chin; C. Goble; A. Rector; D. Chadwick; S. Otenko and Q. Shi, ‘Achieving Fine-grained Access Control in Virtual Organisations’, doi: 10.1002/cpe.v19:9 Concurrency and Computation: Practice and Experience Volume 19 Issue 9, Pages 1333 – 1352

APPENDIX A

Authentication LoA Derivation Algorithms

$ALoA_{WL}$ Algorithm (WL_LoADA.java)

```
package loa_algorithms;

import java.util.Arrays;

/**
 *
 * @author yaol
 */
public class WL_LoADA {

    public WL_LoADA() {
    }
    double Agg_LoA;//[addAttrGroup.length];
    //double Agg_LoAS;
    //int index;

    public double wl_LoA(double addAttrGroup[]) {
        int length = addAttrGroup.length;
        //for (int j = 0; j < length; j++) {
            //for (int i = 0; i < addAttrGroup[j].length; i++) {
            if (length > 1) {
                Arrays.sort(addAttrGroup);
                return addAttrGroup[0];

            } else if (length == 1) {
                Agg_LoA = addAttrGroup[0];
            } else {
                Agg_LoA = -1;
            }
            //}
        //}
        return Agg_LoA;

    }
}
```

ALoAAD-SL Algorithm (SL_LoADA.java)

```
package loa_algorithms;

/**
 *
 * @author yaol
 */
public class SL_LoADA {

    public SL_LoADA() {
    }

    double Agg_LoA;
```

```
    public double add_LoA(double addAttrGroup[]) {
        //int index;

        int length = addAttrGroup.length;

        if (length > 2) {

            Agg_LoA = add_LoASL(addAttrGroup[0], addAttrGroup[1]);

            for (int index = 1; index < addAttrGroup.length - 1;
index++) {
                Agg_LoA = add_LoASL(Agg_LoA, addAttrGroup[index +
1]);
            }

        } else if (length == 2) {
            //add weightings to component LoA value
            Agg_LoA = add_LoASL(addAttrGroup[0], addAttrGroup[1]);

        } else if (length == 1) {
            Agg_LoA = addAttrGroup[0];
        } else {
            Agg_LoA = -1;
        }
        return Agg_LoA;
    }

    protected double add_LoASL(double loa1, double loa2) {
          double uncert1 = 1 - loa1;
          double uncert2 = 1 - loa2;
      double k = uncert1 + uncert2 - uncert1 * uncert2;
      if (k == 0) {
            Agg_LoAS = 1;
      } else {
            Agg_LoAS = (loa1 * uncert2 + loa2 * uncert1) / k;
      }
      return Agg_LoAS;
    }
}
```

ALoAAD-PT Algorithm (PT_LoADA.java)

```
package loa_algorithms;

/**
 *
 * @author yaol
 */
public class PT_LoADA {
    double Agg_LoA;
    public PT_LoADA() {
    }

    public double add_LoA(double addAttrGroup[]) {
        double temp = 1;
        int length = addAttrGroup.length;
        if (length > 1) {
            for (int i = 0; i < length; i++) {
                temp = temp * (1 - addAttrGroup[i]);
            }
            Agg_LoA = 1 - temp;
        } else if (length == 1) {
            Agg_LoA = addAttrGroup[0];
```

```
        } else {
            Agg_LoA = -1;
        }
        return Agg_LoA;
    }
}
```

APPENDIX B

LoA-APDB Database Schema

```
/*
MySQL Data Transfer
Source Host: localhost
Source Database: loa_apdb
Target Host: localhost
Target Database: loa_apdb
Date: 29/05/2010 23:21:27
*/

SET FOREIGN_KEY_CHECKS=0;
-- ----------------------------
-- Table structure for aggregate loa
-- ----------------------------
DROP TABLE IF EXISTS `aggregate loa`;
CREATE TABLE `aggregate loa` (
  `agg_loa_id` int(11) NOT NULL,
  `agg_loa` double NOT NULL,
  `time` datetime NOT NULL,
  PRIMARY KEY (`agg_loa_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ----------------------------
-- Table structure for ahs
-- ----------------------------
DROP TABLE IF EXISTS `ahs`;
CREATE TABLE `ahs` (
  `parent_attr_id` int(5) DEFAULT NULL,
  `child_attr_id` int(5) DEFAULT NULL,
  `left_node` int(5) DEFAULT NULL,
  `right_node` int(5) DEFAULT NULL,
  `relationship` varchar(100) DEFAULT NULL,
  `weighting` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ----------------------------
-- Table structure for attributes
-- ----------------------------
DROP TABLE IF EXISTS `attributes`;
CREATE TABLE `attributes` (
  `attr_ID` int(5) NOT NULL,
  `attr_name` varchar(200) NOT NULL,
  `comp_loa` int(2) DEFAULT NULL,
  PRIMARY KEY (`attr_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ----------------------------
-- Table structure for contributing ahs
-- ----------------------------
DROP TABLE IF EXISTS `contributing ahs`;
CREATE TABLE `contributing ahs` (
```

```
  `hie_id` int(11) NOT NULL,
  `parent_attr_id` int(11) DEFAULT NULL,
  `child_attr_id` int(11) NOT NULL,
  `agg_loa_id` int(11) NOT NULL,
  PRIMARY KEY (`hie_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records
-- ----------------------------
INSERT INTO `aggregate loa` VALUES ('1', '0.71', '2010-05-20
23:15:14');
INSERT INTO `ahs` VALUES (null, '1', '1', '24', 'additive', null);
INSERT INTO `ahs` VALUES ('1', '2', '2', '19', 'additive', '0.8');
INSERT INTO `ahs` VALUES ('1', '3', '20', '23', 'additive', '0.2');
INSERT INTO `ahs` VALUES ('2', '4', '3', '8', 'weakest', '0.33');
INSERT INTO `ahs` VALUES ('2', '5', '9', '18', 'weakest', '0.67');
INSERT INTO `ahs` VALUES ('4', '6', '4', '5', null, null);
INSERT INTO `ahs` VALUES ('4', '7', '6', '7', null, null);
INSERT INTO `ahs` VALUES ('5', '8', '10', '15', 'weakest', null);
INSERT INTO `ahs` VALUES ('5', '9', '16', '17', null, null);
INSERT INTO `ahs` VALUES ('8', '10', '11', '12', null, null);
INSERT INTO `ahs` VALUES ('8', '11', '13', '14', null, null);
INSERT INTO `ahs` VALUES ('3', '19', '21', '22', null, null);
INSERT INTO `attributes` VALUES ('1', 'agg_loa', null);
INSERT INTO `attributes` VALUES ('2', 'id_authN', null);
INSERT INTO `attributes` VALUES ('3', 'location_authN', null);
INSERT INTO `attributes` VALUES ('4', 'password_authN', null);
INSERT INTO `attributes` VALUES ('5', 'certificate_authN', null);
INSERT INTO `attributes` VALUES ('6', 'passwd_authN_token', null);
INSERT INTO `attributes` VALUES ('7', 'passwd_protocol', null);
INSERT INTO `attributes` VALUES ('8', 'cert_authN_token', null);
INSERT INTO `attributes` VALUES ('9', 'cert_protocol', null);
INSERT INTO `attributes` VALUES ('10', 'cert_token_type', null);
INSERT INTO `attributes` VALUES ('11', 'cert_keystore', null);
INSERT INTO `attributes` VALUES ('12', 'passwd', '2');
INSERT INTO `attributes` VALUES ('13', 'ssl', '4');
INSERT INTO `attributes` VALUES ('14', 'id_certificate', '4');
INSERT INTO `attributes` VALUES ('15', 'proxy_certificate', '2');
INSERT INTO `attributes` VALUES ('16', 'cert_soft', '3');
INSERT INTO `attributes` VALUES ('17', 'cert_hard', '4');
INSERT INTO `attributes` VALUES ('19', 'ip', '2');
INSERT INTO `contributing ahs` VALUES ('1', null, '1', '1');
INSERT INTO `contributing ahs` VALUES ('2', '1', '2', '1');
INSERT INTO `contributing ahs` VALUES ('3', '1', '3', '1');
INSERT INTO `contributing ahs` VALUES ('4', '2', '4', '1');
INSERT INTO `contributing ahs` VALUES ('5', '3', '19', '1');
INSERT INTO `contributing ahs` VALUES ('6', '4', '6', '1');
INSERT INTO `contributing ahs` VALUES ('7', '4', '7', '1');
INSERT INTO `contributing ahs` VALUES ('8', '6', '12', '1');
INSERT INTO `contributing ahs` VALUES ('9', '7', '13', '1');
```

APPENDIX C

SAML Assertion Creation method

```
private SAMLAssertion saml() {
        String strIssuer = "EGA-LoADM";
        String strNameID = "password_authN";
        String strNameQualifier = "EGA-LoADM:AS";
        String strNamespace = "urn:bea:security:saml:AC";
        String strAttrName1 = "passwd_authN_token";
        String strAttrName2 = "passwd_protocol";
        SAMLAssertion assertion = null;
        try {
            // Crate the assertion
            assertion = new SAMLAssertion(strIssuer, null, null,
null, null, null);
            // Create the subject
            SAMLSubject subject = new SAMLSubject(new
SAMLNameIdentifier(strNameID, strNameQualifier,
SAMLNameIdentifier.FORMAT_UNSPECIFIED), null, null, null);


subject.addConfirmationMethod(SAMLSubject.CONF_SENDER_VOUCHES);

            // Create the authentication statement
            //Date date = new Date();
            //SAMLAuthenticationStatement authStatement = new
SAMLAuthenticationStatement(subject, LoAAttr, date, null, null,
null);

            //assertion.addStatement(authStatement);

            // Create the attribute statement
            SAMLAttribute attr = new SAMLAttribute(strAttrName1,
strNamespace, null, 0, null);
            SAMLAttribute attr1 = new SAMLAttribute(strAttrName2,
strNamespace, null, 0, null);
            // Here some hardcoded values for the groups attributes
            attr.addValue("passwd");
            attr1.addValue("ssl");

            HashSet set = new HashSet();
            set.add(attr);
            set.add(attr1);
            SAMLSubject subject2 = (SAMLSubject) subject.clone();
            SAMLAttributeStatement attrStatement = new
SAMLAttributeStatement(subject2, set);

            assertion.addStatement(attrStatement);

            SAMLDoNotCacheCondition condition = new
SAMLDoNotCacheCondition();
            assertion.addCondition(condition);

            //System.out.println("AMUserAssertion 1:\n" +
```

```
assertion.toString());
            //SAMLAttributeStatement attr3 = (SAMLAttributeStatement)
assertion.getStatements().next();

//System.out.println(attr3.getAttributes().next().toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return assertion;
    }
```

Weighting Determination code (weightingDet.java)

```
public strictfp class weightingDet
{
  public static void eigen(Complex A[][], Complex lambda[],
                           Complex vec[][], boolean fail[])
  {
    System.out.println("Eigene.eigen(A, lambda, vec, fail)");
    // driver for computing eigenvalues and eigenvectors
    if(A==null || lambda==null || vec==null)
    {
      System.out.println("Error in Eigen2.eigen,"+
                         " null or inconsistent array sizes.");
      return;
    }
    int n = A.length;
    if(A[0].length!=n || vec.length!=n || vec[0].length!=n ||
lambda.length!=n)
    {
      System.out.println("Error in Eigen.eigen,"+
                         " inconsistent array sizes.");
      return;
    }
    fail[0] = false;

    // special cases
    if(n<1) {System.out.println("zero size matrix"); return;}
    int rowcol[] = new int[n];
    Complex B[][] = new Complex[n][n];
    ComplexMatrix.copy(A, B);

    if(n==1)
    {
      lambda[0] = B[0][0];
      vec[0][0] = new Complex(1.0, 0.0);
      return;
    }
    if(n==2)
    {
      twobytwo(B, lambda, vec);
      return;
    }
    System.out.println("calling cxhess");
    cxhess(B, rowcol);
    for(int i=0; i<n; i++) lambda[i] = new Complex(-999.0, -999.0);
    System.out.println("calling cxeig2c");
    cxeig2c(B, lambda, vec, rowcol, fail);
  } // end eigen

  private static void twobytwo(Complex A[][], Complex lambda[],
                               Complex vec[][])
  {
```

```
    Complex b, c, rad, l1, l2;
    Complex Z[] = new Complex[2];
    double t;

    b = A[0][0].add(A[1][1]); // negative
    c = A[0][0].multiply(A[1][1]);
    c = c.subtract(A[0][1].multiply(A[1][0]));
    rad = (b.multiply(b)).subtract(c.multiply(4.0)); // a==1
    rad = rad.sqrt();
    l1 = (b.add(rad)).divide(2.0);
    l2 = (b.subtract(rad)).divide(2.0);
    lambda[0] = l1;
    lambda[1] = l2;
    // eigenvectors in columns
    Z[0] = A[0][1].negate();
    Z[1] = A[0][0].subtract(l1);
    t = ComplexMatrix.norm2(Z);
    vec[0][0] = Z[0].divide(t);
    vec[1][0] = Z[1].divide(t);
    Z[0] = A[1][1].subtract(l2);
    Z[1] = A[1][0].negate();
    t = ComplexMatrix.norm2(Z);
    vec[0][1] = Z[0].divide(t);
    vec[1][1] = Z[1].divide(t);
  }

  private static double sumabs(Complex Z)
  {
    return Math.abs(Z.real())+Math.abs(Z.imaginary());
  } // end sumabs

  private static void cxhess(Complex A[][], int rowcol[])
  {
    int i, k, t;
    Complex x;
    Complex y;

    System.out.println("cxhess(A, rowcol)");
    int n = A.length; // checked before call

    for(int j=0; j<n; j++) rowcol[j] = j;
    k = 0;
    for(int m=k+1; m<n-1; m++) // main reduction loop
    {
      i= m;
      x = new Complex(0.0, 0.0);
      for(int j=m; j<n; j++)
      {
      if(sumabs(A[j][m-1]) > sumabs(x))
        {
          x = A[j][m-1];
          i = j;
        }
      }
      if(i != m)
      {
        //  rowcol row column interchange of H
      t = rowcol[m];
        rowcol[m] = rowcol[i];
        rowcol[i] = t;
      for(int j=m-1; j<n; j++)
```

```
        {
          y = A[i][j];
          A[i][j] = A[m][j];
          A[m][j] = y;
        }
        for(int j=0; j<n; j++) //for J in 1..N loop
        {
          y = A[j][i];
          A[j][i] = A[j][m];
          A[j][m] = y;
        }
      }
      if(sumabs(x) != 0.0)
      {
      for(int ii=m+1; ii<n; ii++)
        {
          y = A[ii][m-1];
          if(sumabs(y) > 0.0)
        {
            y = y.divide(x);
            A[ii][m-1] = y;
            for(int j=m; j<n; j++)
            {
              A[ii][j] = A[ii][j].subtract(y.multiply(A[m][j]));
          }
            for(int j=0; j<n; j++)
            {
              A[j][m] = A[j][m].add(y.multiply(A[j][ii]));
            }
          } // end if
          A[ii][m-1] = new Complex(0.0, 0.0); // just cleanup
        }
      } // end if
    } // end main reduction loop
    System.out.println("result of cxhess=");
    ComplexMatrix.print(A);
    System.out.println("based on rowcol interchanges=");
    Matrix.print(rowcol);
    System.out.println(" ");
  } // end cxhess

  private static void cxeig2c(Complex A[][], Complex lambda[],
                              Complex vec[][],
                              int rowcol[], boolean fail[])
  {
    int j, k, m, mm, low, its, itn, ien;
    double anorm = 0.0;
    double ahr, aahr, acc, xr, xi, yr, yi, zr;
    Complex accnorm;
    Complex x, y, z, yy, T, S;
    int n = A.length; // checked in driver

    low = 0;
    acc = Math.pow(2.0, -23);
    System.out.println("acc="+acc+" = 2^-23");
    T = new Complex(0.0, 0.0);
    itn = 30 * n; // heuristic on maximum iterations
    ComplexMatrix.identity(vec); // initialize to identity Matrix

    // starting from Hessenberg reduction
    for(int ii=n-2; ii>0; ii--) //for i in reverse A'FIRST+1..A'LAST-
```

```
1 loop
    {
      j = rowcol[ii];
      for(k=ii+1; k<n; k++) //for K in i+1..A'LAST loop
      {
        vec[k][ii] = A[k][ii-1];
      }
      if(ii != j)
      {
      for(k=ii; k<n; k++) //for k in i..A'LAST loop
        {
          vec[ii][k] = vec[j][k];
          vec[j][k] = new Complex(0.0, 0.0);
        }
        vec[j][ii] = new Complex(1.0, 0.0);
      }
    }
    ien = n-1; // used as subscript, loop test <=ien

    // ien is decremented
    while(low <= ien) // 260
    {
      System.out.println("260 low="+low+", ien="+ien);
      its = 0;
      // look for small single subdiagonal element
      L280: while(true)  //  280
      {
        System.out.println("in 280");
        k = low;
        // for kk in reverse low+1..ien loop  // 300
        for(int kk=ien; kk>low; kk--) // 300
      {
          System.out.println("300 kk="+kk);

          ahr = sumabs(A[kk][kk-1]);
          aahr = acc * (sumabs(A[kk-1][kk-1]) + sumabs(A[kk][kk]));
          if(ahr <= aahr)
        {
            k = kk;
            break;
          }
        }  //  300
        System.out.println("exiting 300 with k="+k);
        if(k == ien) break L280;  //exit L280 when k = ien;  // 780
        if(itn <= 0)
      {
          fail[0] = true;
          return;
        }

        // compute shift
        if(its == 10 || its == 20)
      {
          S = new Complex(Math.abs(A[ien][ien-1].real()) +
                          Math.abs(A[ien-1][ien-2].real()) ,
                          Math.abs(A[ien][ien-1].imaginary()) +
                          Math.abs(A[ien-1][ien-2].imaginary()));
      }
        else
      {
          S = A[ien][ien];
```

```
          x = A[ien-1][ien].multiply(A[ien][ien-1]);
          if(sumabs(x) > 0.0)
        {
            y = (A[ien-1][ien-1].subtract(S)).divide(new Complex(2.0,
0.0));
            z = ((y.multiply(y)).add(x)).sqrt();
            if(y.real() * z.real() + y.imaginary() * z.imaginary() <
0.0)
          {
              z = z.negate();
            }
            yy = y.add(z);
            S = S.subtract(x.divide(yy));
          } // end if;
        } // end if;  //  400
        for(int i=low; i<=ien; i++) // for i in low..ien loop  // 420
      {
          A[i][i] = A[i][i].subtract(S);
        } // end loop;  //  420
        T = T.add(S);
        its = its + 1;
        itn = itn - 1;
        j = k + 1;

        // look for two consecutive small sub-diagonal elements
        xr = sumabs(A[ien-1][ien-1]);
        yr = sumabs(A[ien][ien-1]);
        zr = sumabs(A[ien][ien]);
        m = k;
        for(mm=ien-1; mm>=j; mm--) // for mm in reverse j..ien-1 loop
// 460
        {
        System.out.println("460 mm="+mm+", m="+m+", j="+j);
          yi = yr;
          yr = sumabs(A[mm][mm-1]);
          xi = zr;
          zr = xr;
          xr = sumabs(A[mm-1][mm-1]);
          if(yr <= (acc * zr/yi *( zr + xr + xi )))
        {
            m = mm;
            break;
          }
        } //end loop;  //  460

        // triangular decomposition  A = L*R
        for(int i=m+1; i<=ien; i++) //for i in m+1..ien loop  // 620
        {
        System.out.println("620 mm="+mm+", m="+m+", i="+i);
          x = A[i-1][i-1];
          y = A[i][i-1];
          if(sumabs(x) >= sumabs(y))
        {
            z = y.divide(x);
            lambda[i] = new Complex(-1.0, 0.0 );
          }
          else
        {
            // interchange rows of A
          for(int jj=i-1; jj<n; jj++) // for j in i-1..n loop  // 540
            {
```

```
              z = A[i-1][jj];
              A[i-1][jj] = A[i][jj];
              A[i][jj] = z;
            } // end loop;  //  540
            z = x.divide(y);
            lambda[i] = new Complex(1.0, 0.0);
          } // end if;
          A[i][i-1] = z;
          for(int jj=i; jj<n; jj++) // for j in i .. N loop  // 600
          {
            A[i][jj] = A[i][jj].subtract(z.multiply(A[i-1][jj]));
          } // end loop;  //  600
        } // end loop;  //  620

        // composition R*L = H
        for(int jj=m+1; jj<=ien; jj++) // for j in m+1..ien loop  //
760
        {
          x = A[jj][jj-1];
          A[jj][jj-1] = new Complex(0.0, 0.0);

          // interchange columns of A and vec if necessary
          if(lambda[jj].real() > 0.0)
          {
            for(int i=low; i<=jj; i++) // for i in low .. j loop  //
660
            {
              z = A[i][jj-1];
              A[i][jj-1] = A[i][jj];
              A[i][jj] = z;
            } // end loop;  //  660
            for(int i=low; i<n; i++) // for i in low .. N loop  //
680
            {
              z = vec[i][jj-1];
              vec[i][jj-1] = vec[i][jj];
              vec[i][jj] = z;
            } // end loop;  //  680
          } // end if

          // end interchange columns
          for(int i=low; i<=jj; i++) // for i in low..j loop  // 720
          {
            A[i][jj-1] = A[i][jj-1].add(x.multiply(A[i][jj]));
          }  //  720
          for(int i=low; i<n; i++) // for i in low..N loop  // 740
          {
            vec[i][jj-1] = vec[i][jj-1].add(x.multiply(vec[i][jj]));
          }  //  740

          // end accumulate transformations
        }  //  760
      }  // 280

      // a root found
      lambda[ien] = A[ien][ien].add(T);
      ien = ien - 1;
    } // end loop;  // 260 while

    // all roots found
    for(int i=0; i<n; i++) // for i in A'RANGE loop
```

```
    {
      anorm = anorm + sumabs(lambda[i]);
      for(int jj=i+1; jj<n; jj++) // for j in i + 1 .. A'LAST loop
      {
        anorm = anorm + sumabs(A[i][jj]);
      }
    }
    accnorm = new Complex(anorm * Math.pow(2.0,-23), 0.0);
    if(anorm == 0.0 || n < 2)
    {
      return;  // done
    }

    // back substitute to set up vec of upper triangular form
    for(ien=n-1; ien>low; ien--) // for ien in reverse low+1..N loop
    {
      x = lambda[ien];
      for(int i=ien-1; i>=low; i--) // for i in reverse low .. ien -
1 loop
      {
        z = A[i][ien];
        for(int jj=i+1; jj<ien; jj++) // for j in i+1..ien-1 loop
        {
          z = z.add(A[i][jj].multiply(A[jj][ien]));
        }
        y = x.subtract(lambda[i]);
        if(sumabs(y) == 0.0)
        {
          y = accnorm;
        }
        A[i][ien] = z.divide(y);
      }
    }

    // multiply by transformation Matrix to give vec of original full
Matrix
    for(int jj=n-1; jj>=0; jj--) // for j in reverse A'RANGE loop
    {
      for(int i=0; i<n; i++) // for i in A'RANGE loop
      {
        z = vec[i][jj];
        for(k=0; k<jj; k++) // for k in A'first..j-1 loop
      {
          z = z.add(vec[i][k].multiply(A[k][jj]));
        }
        vec[i][jj] = z;
      }
    }
  } // end cxeig2c
} // end class Eigen2

// Complex.java
/*
 * Copyright (c) 2003 Jon S. Squire.  All Rights Reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * -Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
```

```
 *
 * -Redistribution in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in
 *  the documentation and/or other materials provided with the
distribution.
 *
 * Neither the name of the author or the names of contributors
 * may be used to endorse or promote products derived from this
software
 * without specific prior written permission.
 *
 * This software is provided "AS IS," without a warranty of any kind.
ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING
 * ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE
 * OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. THE AUTHOR AND
CONTRIBUTORS
 * SHALL NOT BE LIABLE FOR ANY DAMAGES OR LIABILITIES SUFFERED BY
LICENSEE
 * AS A RESULT OF OR RELATING TO USE, MODIFICATION OR DISTRIBUTION OF
THE
 * SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL THE AUTHOR OR
CONTRIBUTORS
 * OR SUCCEEDING LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR
DATA,
 * OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR
PUNITIVE
 * DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY,
 * ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF
THE AUTHOR
 * OR CONTRIBUTORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.
 *
 * You acknowledge that this software is not designed, licensed or
 * intended for use in the design, construction, operation or
 * maintenance of any human use medical device.
 */


/** Immutable, complex numbers. A Complex consists of a real
 *  and imaginary part, called Cartesian coordinates.
 *
 *  The Complex class provides methods for arithmetic such as:
 *  add, subtract, multiply, divide, negate and invert.
 *  Also provided are complex functions sin, cos, tan, asin, acos,
atan,
 *  sqrt, log, exp, pow, sinh, cosh, tanh, atanh.
 *
 *  Source code <a href="Complex.java">Complex.java</a>
 */

strictfp class Complex extends Object
{
  double x, y; // Cartesian representation of complex

  /** cartesian coordinates real and imaginary are NaN */
  public Complex(){x=Double.NaN; y=Double.NaN;}

  /** construct a copy of a Complex object */
```

```
  public Complex(Complex z){x=z.real(); y=z.imaginary();}

  /** real value, imaginary=0.0 */
  public Complex(double x){this.x=x; y=0.0;}

  /** cartesian coordinates real and imaginary */
  public Complex(double x, double y){this.x=x; this.y=y;}

  /** convert cartesian to polar */
  public Complex polar(){double r = StrictMath.sqrt(
                                    this.x*this.x+this.y*this.y);
                         double a = StrictMath.atan2(this.y,this.x);
                         return new Complex(r,a);}

  /** convert polar to cartesian */
  public Complex cartesian(){return new
Complex(this.x*StrictMath.cos(this.y),

this.x*StrictMath.sin(this.y));}

  /** extract the real part of the complex number */
  public double real(){return this.x;}

  /** extract the imaginary part of the complex number */
  public double imaginary(){return this.y;}

  /** extract the magnitude of the complex number */
  public double magnitude(){return

StrictMath.sqrt(this.x*this.x+this.y*this.y);}

  /** extract the argument of the complex number */
  public double argument(){return StrictMath.atan2(this.y,this.x);}

  /** add complex numbers */
  public Complex add(Complex z){return new Complex
                                (this.x+z.x, this.y+z.y);}

  /** add a double to a complex number */
  public Complex add(double d){return new Complex
                                (this.x+d, this.y);}

  /** subtract z from the complex number */
  public Complex subtract(Complex z){return new Complex
                                     (this.x-z.x, this.y-z.y);}

  /** subtract the double d from the complex number */
  public Complex subtract(double d){return new Complex
                                     (this.x-d, this.y);}

  /** negate the complex number */
  public Complex negate(){return new Complex(-this.x, -this.y);}

  /** multiply complex numbers */
  public Complex multiply(Complex z){return new Complex
                          (this.x*z.x-this.y*z.y,
                           this.x*z.y+this.y*z.x);}

  /** multiply a complex number by a double */
  public Complex multiply(double d){return new
Complex(this.x*d,this.y*d);}
```

```
  /** divide the complex number by z */
  public Complex divide(Complex z){double r=z.x*z.x+z.y*z.y;
                                   return new Complex
                                   ((this.x*z.x+this.y*z.y)/r,
                                    (this.y*z.x-this.x*z.y)/r);}

  /** divide the complex number by the double d */
  public Complex divide(double d){return new
Complex(this.x/d,this.y/d);}

  /** invert the complex number */
  public Complex invert(){double r=this.x*this.x+this.y*this.y;
                          return new Complex(this.x/r, -this.y/r);}

  /** conjugate the complex number */
  public Complex conjugate(){return new Complex(this.x, -this.y);}

  /** compute the absolute value of a complex number */
  public double abs(){return
StrictMath.sqrt(this.x*this.x+this.y*this.y);}

  /** compare complex numbers for equality */
  public boolean equals(Complex z){return (z.x==this.x) &&
                                          (z.y==this.y);}

  /** convert a complex number to a String.
   *  Complex z = new Complex(1.0,2.0);
   *  System.out.println("z="+z); */
  public String toString(){return new
String("("+this.x+","+this.y+")");}

  /** convert text representation to a Complex.
   *  input format  (real_double,imaginary_double) */
  public static Complex parseComplex(String s){
      int from = s.indexOf('(');
      if(from==-1) return null;
      int to = s.indexOf(',',from);
      double x = Double.parseDouble(s.substring(from+1,to));
      from = to;
      to = s.indexOf(')',from);
      double y = Double.parseDouble(s.substring(from+1,to));
      return new Complex(x,y); }

  /** compute e to the power of the complex number */
  public Complex exp(){double exp_x=StrictMath.exp(this.x);
                       return new Complex
                       (exp_x*StrictMath.cos(this.y),
                        exp_x*StrictMath.sin(this.y));}

  /** compute the natural logarithm of the complex number */
  public Complex log(){double rpart=StrictMath.sqrt(
                       this.x*this.x+this.y*this.y);
                       double ipart=StrictMath.atan2(this.y,this.x);
                       if(ipart>StrictMath.PI) ipart=ipart-
2.0*StrictMath.PI;
                       return new Complex(StrictMath.log(rpart),
ipart);}

  /** compute the square root of the complex number */
  public Complex sqrt(){double
```

```
r=StrictMath.sqrt(this.x*this.x+this.y*this.y);
                        double rpart=StrictMath.sqrt(0.5*(r+this.x));
                        double ipart=StrictMath.sqrt(0.5*(r-this.x));
                        if(this.y<0.0) ipart=-ipart;
                        return new Complex(rpart,ipart);}

  /** compute the complex number raised to the power z */
  public Complex pow(Complex z){Complex a=z.multiply(this.log());
                                return a.exp();}

  /** compute the complex number raised to the power double d */
  public Complex pow(double d){Complex a=(this.log()).multiply(d);
                                return a.exp();}

  /** compute the sin of the complex number */
  public Complex sin(){return new Complex
                       (StrictMath.sin(this.x)*cosh(this.y),
                        StrictMath.cos(this.x)*sinh(this.y));}

  /** compute the cosine of the complex number */
  public Complex cos(){return new Complex
                       (StrictMath.cos(this.x)*cosh(this.y),
                        -StrictMath.sin(this.x)*sinh(this.y));}

  /** compute the tangent of the complex number */
  public Complex tan(){return (this.sin()).divide(this.cos());}

  /** compute the arcsine of a complex number */
  public Complex asin(){Complex IM = new Complex(0.0,-1.0);
                      Complex ZP = this.multiply(IM);
                      Complex ZM = (new Complex(1.0,0.0)).subtract

(this.multiply(this)).sqrt().add(ZP);
                      return ZM.log().multiply(new
Complex(0.0,1.0));}

  /** compute the arccosine of a complex number */
  public Complex acos(){Complex IM = new Complex(0.0,-1.0);
                      Complex ZM = (new Complex(1.0,0.0)).subtract

(this.multiply(this)).sqrt().multiply
                                        (IM).add(this);
                      return ZM.log().multiply(new
Complex(0.0,1.0));}

  /** compute the arctangent of a complex number */
  public Complex atan(){Complex IM = new Complex(0.0,-1.0);
                        Complex ZP = new Complex(this.x,this.y-1.0);
                        Complex ZM = new Complex(-this.x,-this.y-
1.0);
                        return
IM.multiply(ZP.divide(ZM).log()).divide(2.0);}

  /** compute the hyperbolic sin of the complex number */
  public Complex sinh(){return new Complex
                       (sinh(this.x)*StrictMath.cos(this.y),
                        cosh(this.x)*StrictMath.sin(this.y));}

  /** compute the hyperbolic cosine of the complex number */
  public Complex cosh(){return new Complex
                       (cosh(this.x)*StrictMath.cos(this.y),
```

```
                        sinh(this.x)*StrictMath.sin(this.y));}

  /** compute the hyperbolic tangent of the complex number */
  public Complex tanh(){return (this.sinh()).divide(this.cosh());}

  /** compute the inverse hyperbolic tangent of a complex number */
    public Complex atanh(){return (((this.add(1.0)).log()).subtract(
                           ((this.subtract(1.0)).negate()).log())
                           .divide(2.0));}


  // local - should be a good implementation in StrictMath
  private double sinh(double x){return(
                                StrictMath.exp(x)-StrictMath.exp(-
x))/2.0;}
  private double cosh(double x){return(
                                StrictMath.exp(x)+StrictMath.exp(-
x))/2.0;}
}
```

LoA-ACM module (loa_acm.java)

```
package ehealth.security.LoA;
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Hashtable;
import java.util.Iterator;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.opensaml.SAMLAssertion;
import org.opensaml.SAMLAttribute;
import org.opensaml.SAMLAttributeStatement;

/**
 *
 * @author yaol
 */
public class loa_acm extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            HttpSession session = request.getSession(true);
            SAMLAssertion assertion = (SAMLAssertion)
```

```
session.getAttribute("saml");

            SAMLAttributeStatement samlAttrState =
(SAMLAttributeStatement) assertion.getStatements().next();
            Iterator samlAttrs = samlAttrState.getAttributes();

            Hashtable contributing_attr = new Hashtable();
            SAMLAttribute samlAttr = null;
            while (samlAttrs.hasNext()) {
                samlAttr = (SAMLAttribute) samlAttrs.next();
                contributing_attr.put(samlAttr.getName(),
samlAttr.getValues().next().toString());
            }
            session.setAttribute("contributing_Attributes",
contributing_attr);

response.sendRedirect(response.encodeRedirectURL("aloa_dm"));
        } finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet
methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```